UNIVERSIDAD NACIONAL DE PIURA

Facultad de Ingeniería Industrial

Escuela Profesional de Ingeniería Informática

TESIS

“ANÁLISIS COMPARATIVO DE LOS TIEMPOS DE RESPUESTA DE UN MÓDULO WEB DE TRÁMITE DOCUMENTARIO DESARROLLADO CON LOS FRAMEWORKS SPRING Y STRUTS 2”

Presentada por:

Luis Alberto Sandoval Arévalo

PARA OPTAR EL TÍTULO PROFESIONAL DE
Ingeniero Informático

Línea de investigación: Computación – Ingeniería de Software

Piura, Perú

2018

UNIVERSIDAD NACIONAL DE PIURA

Facultad de Ingeniería Industrial

Escuela Profesional de Ingeniería Informática

TESIS

"ANÁLISIS COMPARATIVO DE LOS TIEMPOS DE RESPUESTA DE UN MÓDULO WEB DE TRÁMITE DOCUMENTARIO DESARROLLADO CON LOS FRAMEWORKS SPRING Y STRUTS 2"

Presentado por:

Luis Alberto Sandoval Arévalo

Asesorado por:

MSc. Víctor Hugo Valle Ríos

Co-asesorado por:

Dr. Moisés David Saavedra Arango

Línea de investigación: Computación – Ingeniería de Software

DECLARACIÓN JURADA DE ORIGINALIDAD DE LA TESIS

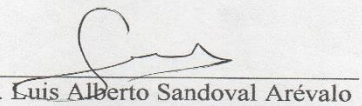
Yo, LUIS ALBERTO SANDOVAL ARÉVALO identificado con CU N° 0512015068, DNI Nº 47081541, Bachiller de Escuela Profesional de Ingeniería Informática, de la Facultad de Ingeniería Industrial y domiciliado en la Urbanización 21 de Agosto manzana C lote 6 del Distrito de Piura, Provincia de Piura, Departamento de Piura. Con número de celular: 997691486 e Email: lsandoval0000@gmail.com.

"ANÁLISIS COMPARATIVO DE LOS TIEMPOS DE RESPUESTA DE UN MÓDULO WEB DE TRÁMITE DOCUMENTARIO DESARROLLADO CON LOS FRAMEWORKS SPRING Y STRUTS 2"

DECLARO BAJO JURAMENTO: que la tesis que presento es original e inédita, no siendo copia parcial ni total de una tesis desarrollada, y/o realizada en el Perú o en el Extranjero, en caso contrario de resultar falsa la información que proporciono, me sujeto a los alcances de lo establecido en el Art. Nº 411, del código Penal concordante con el Art. 32º de la Ley Nº 27444, y Ley del Procedimiento Administrativo General y las Normas Legales de Protección a los Derechos de Autor.

En fe de lo cual firmo la presente.

Piura 15 mayo del 2019



Bach. Luis Alberto Sandoval Arévalo

DNI N° 47081541

Artículo 411.- El que, en un procedimiento administrativo, hace una falsa declaración en relación con hechos o circunstancias que le corresponde probar, violando la presunción de veracidad establecida por ley, será reprimido con pena privativa de libertad no menor de uno ni mayor de cuatro años.

Art. 4. Inciso 4.12 del Reglamento del Registro Nacional de Trabajos de Investigación para optar grados académicos y títulos profesionales –RENATI Resolución de Consejo Directivo Nº 033-2016-SUNEDU/CD

UNIVERSIDAD NACIONAL DE PIURA

Facultad de Ingeniería Industrial

Escuela Profesional de Ingeniería Informática

TESIS

"ANÁLISIS COMPARATIVO DE LOS TIEMPOS DE RESPUESTA DE UN MÓDULO WEB DE TRÁMITE DOCUMENTARIO DESARROLLADO CON LOS FRAMEWORKS SPRING Y STRUTS 2"

Presidente: _____

Dr. Pedro Criollo Gonzales

Secretario: _____

MSc. Yolanda Esther Lizana Puelles

Vocal: _____

MSc. Héctor Wilmer Fiestas Bancayán

Línea de investigación: Computación – Ingeniería de Software

UNIVERSIDAD NACIONAL DE PIURA
FACULTAD DE INGENIERÍA INDUSTRIAL
DECANATO

ACTA DE EVALUACIÓN Y SUSTENTACIÓN DE TESIS

Expediente N° 1548 / 2017

Los miembros del Jurado Calificador Ad-Hoc de la Sustentación de Tesis nombrado con Resolución N° **029-CF-FII-UNP-18** de fecha 11/01/2018 que suscriben, se reunieron en acto público en la sala de exposiciones de la Facultad de Ingeniería Industrial de la Universidad Nacional de Piura, el día **10 de Mayo del 2019** a las **10:00 am**, para evaluar la defensa de la Tesis titulada **"ANÁLISIS COMPARATIVO DE LOS TIEMPOS DE RESPUESTA DE UN MÓDULO WEB DE TRÁMITE DOCUMENTARIO DESARROLLADO CON LOS FRAMEWORKS SPRING Y STRUTS 2"**, presentada por el Bachiller **LUIS ALBERTO SANDOVAL ARÉVALO** y asesorado por el **MSc. VÍCTOR HUGO VALLE RÍOS** y co-asesorado por el **Dr. MOISÉS DAVID SAAVEDRA ARANGO**.
Después de haber calificado el Informe Final de la Tesis, escuchada la sustentación y las respuestas a las preguntas formuladas por el Jurado, se le declara ..APROBADO... para optar el Título de **INGENIERO INFORMÁTICO** con el puntaje de ..86.. que corresponde al calificativo de SOBRESALIENTE

Jurado / Calificación	Presidente	Secretario	Vocal	Puntaje Promedio
Documento (Max 60 puntos)	54	54	43	50.3
Sustentación (Max 40 puntos)	36	35	36	35.9
PUNTAJE TOTAL				86.2

En consecuencia, el sustentante queda en condición de recibir el Título Profesional que se indica, conferido por el Consejo Universitario de la Universidad Nacional de Piura de conformidad con las normas Estatutarias y la Ley Universitaria en vigencia.

FACULTAD DE INGENIERÍA INDUSTRIAL
Dirección de Investigación
* UNP *

Ciudad Universitaria, 10 de Mayo del 2019

Dr. PEDRO ANTONIO CRIOLLO GONZALES	MSc. ESTHER YOLANDA LIZANA PUELLES	MSc. HECTOR WILMER FIESTAS BANCAYÁN
PRESIDENTE	**SECRETARIO**	**VOCAL**

DEDICATORIA

Dedico el presente trabajo a mis padres, hermanos y amigos por ser mi motivo de perseverancia y superación constante.

Por siempre darme su apoyo constante a lo largo de mi vida universitaria, brindándome los consejos a seguir y forjándome siempre por el camino del bien.

AGRADECIMIENTOS

Agradezco a mis padres y hermanos por estar junto a mí incondicionalmente.

A mis amigos que siempre han confiado y creído en mí.

A mi asesor de tesis Ing. Víctor Valle Ríos y co-asesor Dr. Moisés Saavedra Arango por la guía brindada durante el desarrollo del proyecto.

A todos aquellos que directa o indirectamente fueron parte de este proceso brindándome su conocimiento y mejores deseos.

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE ANEXOS

RESUMEN

Los usuarios siempre están a la expectativa de poder usar sin contratiempos los sistemas Web, estos tienen su carta de presentación principal ante los usuarios, en la velocidad con que cargan las páginas que están o serán usadas, por lo tanto, este tiempo es vital y afecta positiva o negativamente en los usuarios. En la actualidad el lenguaje de programación JAVA se encuentra liderando los *rankings* de popularidad, al igual que los *frameworks back-end* para *Java Enterprise Edition* (JEE), Spring y Struts *2*.

El presente trabajo de investigación tiene por objetivo analizar comparativamente los tiempos de respuesta de un módulo Web de trámite documentario desarrollado con los *frameworks* Spring en su versión 5 y Struts *2* con el fin de poder determinar si existe una diferencia significativa, desde el punto de vista estadístico, en los tiempos de respuesta, presentando un enfoque cualitativo con un diseño de investigación experimental empleado un diseño factorial 2x3x3, presentando como factores: *framework de lado servidor*, *tipo de operación* y *número de usuarios concurrentes*. El análisis de los datos es realizado mediante dos pruebas estadísticas, el análisis de varianza factorial – ANOVA y la prueba de diferencia de medias estadísticas, previo a ello, se emplea el software *JMeter* para la creación de planes de prueba y recolección de los tiempos de respuesta como resultado de la ejecución de las pruebas.

Los resultados que se obtienen de las pruebas estadísticas muestran que existe una diferencia significativa en los tiempos de respuesta de ambos *frameworks*, siendo el *framework* Spring el que genera el menor tiempo de respuesta y el factor predominante que genera esta diferencia el *número de usuarios concurrentes*. Teniendo en cuenta los resultados obtenidos se recomienda, el uso del *framework* Spring en el desarrollo de aplicaciones Web en el entorno *Java Enterprise Edition* (JEE).

Como recomendación, se motiva a realizar investigaciones adicionales sobre los tiempos de respuesta máximos y mínimos de ambos *frameworks* para determinar un factor de rendimiento, esto sería de gran utilidad, pues sería un complemento a la presente investigación, permitiendo tomar con un grado mayor de certeza decisiones en el desarrollo de futuras aplicaciones Web.

Palabras clave: Spring, Struts 2, análisis comparativo, ANOVA.

ABSTRACT

Users are always expecting to be able to use Web systems without problems, they have their main presentation to the users, in the speed they load the pages that are or will be used, therefore, this time is vital and affects users positively or negatively. Currently the JAVA programming language is leading the popularity rankings, also their back-end frameworks for Java Enterprise Edition (JEE), Spring and Struts 2.

The objective of this research work is to analyze comparatively the response times of a documentary processing Web module developed with Spring frameworks version 5 and Struts 2 in order to determine if there is a significant difference, from the point of statistical view, in the response times, presenting a qualitative approach with an experimental design using a 2x3x3 factorial design, with the following factors: server-side framework, operation type and number of concurrent users. The analysis of the data was made with the use of two statistical tests, the analysis of factorial variance - ANOVA and the test of difference of statistical means, prior to that, the JMeter software is used for the creation of test plans and collection of the response times as a result of the execution of the tests.

The results obtained from the statistical tests showed that there is a significant difference in the response times of both frameworks, with the Spring framework generating the shortest response time and the predominant factor that generates this difference the number of concurrent users. Taking into account the results obtained, it is recommended to use the Spring framework in the development of Web applications in the Java Enterprise Edition (JEE) environment.

As a recommendation, encouraging additional research on the maximum and minimum response times of both frameworks to determine a performance factor, this would be very useful, since it would be a complement to the present investigation, allowing to take decisions with a greater degree of certainty in the development of future Web applications.

Keywords: Spring, Struts 2, comparative analysis, ANOVA.

INTRODUCCIÓN

Everts (2015) en uno de sus artículos Web, resume los resultados obtenidos de un estudio realizado en el año 2010 por *TRAC Research* a más de 300 empresas internacionales, el cual tenía por objetivo determinar el impacto real de las interrupciones de servicio Web versus tiempos de respuesta elevados en las aplicaciones Web, determinando que en promedio las empresas registran $21,000 de pérdidas en sus ingresos por una hora de interrupción de servicio; 4.4 segundos era el retraso en los tiempos de respuesta a partir de los cuales una empresa empezaba a registrar pérdidas en sus ingresos por un monto promedio de $4,100 por hora, pero se determinó que la presencia de la lentitud en las aplicaciones Web era diez veces más frecuentes que las interrupciones de servicio, generando cuantiosas pérdidas en un mes contable.

Walmart Labs (2012) llevó un estudio en su propio sitio Web, el cual presentaba tiempos de respuesta elevados, determinando que por cada un segundo de mejora en los tiempos de respuesta, el sitio experimentaba hasta un 2% de incremento en su tasa de conversión y por cada 100 milisegundos sus ingresos incrementaban hasta en 1%.

Observando la problemática generada por los tiempos de respuesta y su impacto no solo en factores monetarios, sino también en los usuarios de las aplicaciones Web, se plantea la realización de un análisis comparativo entre los tiempos de respuesta, tomando como caso de estudio el proceso de trámite documentario realizado en la región de Piura, desarrollando un módulo Web con los *frameworks* back-end de *Java Enterprise Edition*, Spring y Struts 2, ambos por mantener una presencia significativa en el mercado de desarrollo de aplicaciones Web, con la finalidad de determinar la mejor opción a elegir entre los frameworks *back-end* Java más populares para la versión *Enterprise Edition* (JEE) que permita manejar tiempos de respuesta aceptables y a su vez generar documentación actualizada y fiable que puede ser usada como guía o referencia no solo por profesionales, sino también por alumnos y docentes de los diferentes centros de formación, garantizando un conocimiento más profundo respecto a los *frameworks* y su impacto en los tiempos de respuesta al ser usados en el desarrollo de aplicaciones Web.

En el capítulo uno se describe de forma detallada la realidad problemática, se plantea el problema principal, el alcance así como la importancia y justificación para la realización de la presente investigación.

En el capítulo dos se presentan las bases teóricas que son el sustento para la realización de la presente investigación, se incluyen investigaciones que sirven de guía para la realización del análisis de los datos, y se plantea la hipótesis general acorde al tema de investigación.

En el capítulo tres se presenta el marco metodológico, donde se define el tipo de investigación, los sujetos de la investigación, técnicas de recolección de datos, así como la realización de los objetivos específicos planteados.

En el capítulo cuatro se presenta la descripción de los resultados obtenidos, empleado pruebas estadísticas para el análisis de los datos y la contrastación de hipótesis, así como la discusión de los resultados obtenidos.

Finalmente se presentan las conclusiones y recomendaciones que se obtuvieron al realizar la presente investigación, así como los anexos que se consideran relevantes para la investigación y bibliografía consultada.

I. ASPECTOS DE LA PROBLEMÁTICA

1.1 Descripción de la realidad problemática

Los usuarios siempre están a la expectativa de poder usar sin contratiempos los sistemas Web, estos sistemas tienen su carta de presentación principal ante los usuarios, en la velocidad con que cargan las páginas que están o serán usadas, por lo tanto, este tiempo es vital y afecta positiva o negativamente en los usuarios. Según un estudio realizado por Akamai Technologies, Inc (2009):

> Según los comentarios de 1.048 compradores en línea encuestados, el 47 por ciento de los consumidores esperan que una página Web se cargue en dos segundos o menos, el 40 por ciento de los consumidores esperarán no más de tres segundos para que una página Web cargue antes de abandonar el sitio. El 52 por ciento de los compradores en línea afirmaron que la carga rápida de la página es importante para la lealtad de su sitio, el 14 por ciento comenzará a comprar en otro sitio.

Ohye (2010) indicó: "dos segundos es el umbral para la aceptabilidad del sitio Web de comercio electrónico. En Google, apuntamos a menos de medio segundo". Esto refleja la necesidad que tienen las empresas de mantener los tiempos de respuesta de sus aplicaciones Web, lo más bajos posibles, con el fin de evitar una fuga importante de clientes, lo cual se reflejaría negativamente en la empresa.

En la actualidad, Java es el lenguaje de programación más popular del mercado según el rango de popularidad de TIOBE (**ver ANEXO N° 01**), la mayor parte de los desarrolladores de *software* que usan este lenguaje, se han abocado a emplear *frameworks back-end*, estos *frameworks* son empleados según la necesidad de los empleadores, sin antes dar tiempo a realizar un análisis de cómo afecta a los tiempos de respuesta el utilizar estas herramientas en el desarrollo de *software* o en el producto final, porque ello demandaría tiempo y recursos, con los cuales, en la mayoría de los casos, no se cuenta. El uso de *frameworks back-end* podría afectar al desarrollo de las aplicaciones, en el caso de las aplicaciones Web esto se podría apreciar en los tiempos de respuesta, además de ello, cada *framework back-end* tiene un tiempo de evolución variable, cada versión posee cambios en sus características acorde con el rumbo al cual se está enfocando el *framework*, un claro ejemplo de ello es Spring Framework, el cual presenta cambios significativos entre sus versiones 3.x.x, 4.x.x y 5.x.x, como la separación en módulos de sus funciones y la aparición de nuevas herramientas, lo cual ha traído consigo varios problemas, tales como el cambio en la configuración necesaria para utilizar un determinado módulo o la incorporación de nuevas formas de desarrollo utilizando las mismas o nuevas herramientas que posee la nueva versión del *framework*. En contraste con el impacto que los *frameworks back-end* pueden tener en los tiempos de respuesta de una aplicación Web, los *frameworks front-end* agregan el tiempo que toman en procesar y mostrar la respuesta dada desde el servidor al usuario al tiempo de respuesta, por lo general, este tiempo tiende a ser disminuido drásticamente simplificando las interfaces con las que interactúa el usuario, dejando como prioridad principal el tiempo que agrega el procesamiento de las peticiones en el lado del servidor, donde los *frameworks back-end* son usados, y donde se debe tener especial cuidado, puesto que con ellos se realizan los procesos que conllevan las aplicaciones Web, si estos procesos son programados de forma errónea, ya sea mediante una mala codificación, la configuración o uso incorrecto de un módulo de los *frameworks* en uso, etc., puede llevar a incrementar drásticamente el tiempo de respuesta.

Es por ello que se realizó un análisis comparativo entre los *frameworks back-end* Spring y Struts 2. Según TIOBE (2018): "actualmente tanto Spring como Struts se encuentran liderando los *rankings* de popularidad de *frameworks* para desarrollo de *software*". Ambos usan el diseño modelo – vista – controlador (MVC) implementados desde la perspectiva de sus respectivos creadores, Ramírez (2017) determinó, en base a su investigación, que el módulo Web que más se desarrolla en la ciudad de Piura, es el módulo de trámite documentario, efectuándose el análisis comparativo en el mismo lugar en el que

se desarrolló la investigación, con el fin de obtener datos específicos y adecuados a la realidad tecnológica.

1.1.1 Formulación y planteamiento del problema de investigación

a. Pregunta general

¿Cómo se analiza comparativamente los tiempos de respuesta de un módulo Web de trámite documentario desarrollado con los *frameworks* Spring y Struts 2?

b. Preguntas específicas

- ¿Cuáles serán los parámetros necesarios para el desarrollo del módulo Web según la metodología RUP?
- ¿Cómo medir el tiempo de respuesta del módulo Web de trámite documentario desarrollado con el *framework* Spring?
- ¿Cómo medir el tiempo de respuesta del módulo Web de trámite documentario desarrollado con el *framework* Struts 2?

1.2 Justificación e importancia de la investigación

1.2.1 Justificación

Con el avance de la tecnología, surgen técnicas y herramientas que aceleran, mejoran y simplifican los procesos inmersos en el desarrollo de *software*, esta tecnología es usada por las empresas para incrementar su productividad, buscando obtener el mayor rendimiento tanto de su personal como del *software* que desarrollan. La decisión de haber realizado este análisis comparativo, se debió a que la mayoría de organizaciones y desarrolladores suelen basarse en la popularidad y/o fuentes no confiables para elegir el o los *frameworks* que serán usados para el desarrollo del *software*, es por ello que con esta investigación se determinó la mejor opción a elegir entre los *frameworks back-end* Java más populares para la versión *Enterprise Edition* (JEE) que permita manejar tiempos de respuesta aceptables. Además, los *frameworks* están en constante evolución, por lo cual, investigaciones como la presente son necesarias para encontrar las mejores opciones de desarrollo y lograr obtener *software* de calidad.

1.2.2 Importancia

Con el estudio comparativo se busca facilitar la decisión de usar un determinado *framework* para el desarrollo de *software*, esto debido a que cada uno de ellos tiene sus particularidades. Esta investigación es de gran importancia, ya que su desarrollo permitió obtener documentación actualizada y de confianza, la misma que puede ser usada como guía o referencia no solo por profesionales, sino también por alumnos y docentes de los diferentes centros de formación, garantizando un conocimiento más profundo respecto a los *frameworks* y su impacto en los tiempos de respuesta al ser usados en el desarrollo de aplicaciones Web. Además, servirá como motivación para estudiantes interesados en investigar las tecnologías nacientes, aportará nuevos conocimientos y permitió evaluar qué tan flexible puede llegar a ser un *framework* al ser usado en el desarrollo de aplicaciones.

1.3 Objetivos

1.3.1 Objetivo general

Analizar comparativamente los tiempos de respuesta de un módulo Web de trámite documentario desarrollado con los *frameworks* Spring y Struts 2.

1.3.2 Objetivos específicos

- Determinar los parámetros necesarios para el desarrollo del módulo Web según la metodología RUP.
- Medir el tiempo de respuesta del módulo Web de trámite documentario desarrollado con el *framework* Spring.
- Medir el tiempo de respuesta del módulo Web de trámite documentario desarrollado con el *framework* Struts 2.

1.4 Delimitación de la investigación

La presente investigación fue aplicada a los *frameworks* Spring en su versión 5.0.4 y Struts 2 en su versión 2.5.16, utilizando como caso de estudio el proceso de trámite documentario que se realiza en la región de Piura.

II. MARCO TEÓRICO

2.1 Antecedentes de la investigación

Ramírez (2017) realizó la investigación *Análisis comparativo del tiempo de ejecución de una Aplicación Web desarrollada en diferentes marcos de trabajo utilizados en el lado del cliente y en el lado del servidor*, desarrollado para obtener el título profesional de Ingeniero Informático en la Universidad Nacional de Piura - Perú, teniendo como objetivo realizar un análisis comparativo de los tiempos de ejecución de una aplicación Web desarrollada en diferentes marcos de trabajo utilizados en el lado del cliente y en el lado del servidor, utilizando la metodología *Rational Unified Process* (RUP), teniendo como suposición que existe diferencia significativa en el tiempo de ejecución de una aplicación Web desarrollada en diferentes marcos de trabajo utilizados en el lado del cliente y en el lado del servidor, llegando a la conclusión que existe diferencia significativa de los tiempos de ejecución de una aplicación Web utilizando diferentes marcos de trabajo, cada vez que esta trabaje con cantidades considerables de información, para el desarrollo de aplicaciones donde el tiempo de ejecución es un factor importante, el *framework* Laravel utilizado de manera conjunta con el *framework* AngularJS es la mejor opción para el desarrollo de cualquier aplicación Web; recomendando que para el lado cliente se utilice AngularJS y para el lado servidor Laravel debido a que ambos son eficientes y permiten un desarrollo sencillo y que los análisis de los *frameworks* debían tomar en cuenta otros factores, como por ejemplo, la seguridad. El aporte de la investigación son los datos del desarrollo de módulos Web en la región Piura, así como la base para llevar a cabo el análisis comparativo entre *frameworks*.

Ruiz (2011) realizó la investigación *Comparativa entre el desarrollo Web usando el framework Jboss Seam y el desarrollo tradicional*, desarrollado para obtener el título profesional de Ingeniero Industrial y de Sistemas en la Universidad de Piura - Perú, teniendo como objetivo comparar los costos y los tiempos del desarrollo de manera tradicional y del desarrollo con el *framework* Jboss Seam, así como definir las consideraciones a tener en cuenta antes de optar por usarlo para el desarrollo de una aplicación, utilizando *Component-Based Development* (Desarrollo Basado en Componentes - CBD), llegando a la conclusión que Seam es un *framework* de desarrollo para la construcción de aplicaciones de nueva generación Web 2.0, unifica tecnologías tales como Ajax, *JavaServer Faces* (JSF) y *Enterprise JavaBeans* (EJB) para que funcionen en conjunto y faciliten el desarrollo ágil de aplicaciones de alta interacción con el usuario; recomendando que si no se cuenta con conocimientos de Seam y la aplicación que se quiere desarrollar es de baja escala, teniendo en cuenta que el periodo de aprendizaje estimado es de 2 meses, es necesario evaluar si el tiempo que tomaría desarrollarla sin un *framework* podría ser menor al tiempo que tomará aprender y dominar Seam para luego poder programar la aplicación. El aporte de la investigación es un modelo comparativo por factores, permitiendo obtener información valiosa, para así determinar cuál forma de desarrollo de aplicaciones sería la indicada entre JBoss Seam y el desarrollo tradicional de aplicaciones.

Gonzales y Tarifeño (2016), realizaron la investigación *Análisis comparativo de frameworks de arquitectura empresarial para el alineamiento estratégico de tecnologías de información*, desarrollado para obtener el título profesional de Ingeniero de Sistemas en la Universidad Señor de Sipán - Perú, teniendo como objetivo analizar comparativamente frameworks de arquitectura empresarial para identificar cuál de ellos resultaría más beneficioso de aplicar según resultados de los criterios comparativos planteados, utilizando las arquitecturas '*The Open Group Architecture Framework*' (TOGAF), Zachman y '*Department of Defense Architecture Framework*' (DoDAF), llegando a la conclusión que '*The Open Group Architecture Framework*' (TOGAF) es un método más práctico para su creación, ya que permite establecer una arquitectura más sólida, al tener sus herramientas bien definidas; todo esto hace que su implementación sea más rápida, disminuyendo los costos; recomendando que las empresas inicien su arquitectura empresarial basada en uno solo, conforme vayan avanzando en la aplicación de este pueden utilizar ciertas características de otro *framework* de Arquitectura Empresarial para complementarlo. El aporte de la investigación es un análisis comparativo,

pero bajo los criterios establecidos por ObjectWatch Inc, un planteamiento de análisis basado en la experiencia de profesionales en tecnologías de la información.

Guerrero (2016) realizó la investigación *Estudio comparativo de los frameworks Ruby on Rails y Django para la implementación de un sistema informático de control y administración de network marketing*, desarrollado para obtener el título profesional Ingeniero en Ciencias Computacionales en la Universidad Técnica del Norte - Ecuador, teniendo como objetivo establecer un estudio comparativo entre los *frameworks* Ruby On Rails y Django para implementar un sistema de control y administración de *network marketing*, utilizando la metodología *Object Oriented Hypermedia Design Model* (Modelo de Diseño de Hipermedia Orientado a Objetos - OOHDM), llegando a la conclusión que en base al estudio comparativo realizado, Djando resultó ser el *framework* adecuado para la implementación del sistema informático de control y administración de *network marketing* puesto que obtuvo una mayor puntuación en base a los parámetros observados; recomendando que cada *framework* tiene sus propios ámbitos de aplicación, para lo cual, se debería siempre realizar un análisis comparativo previo, para así determinar la mejor opción entre herramientas disponibles. El aporte de la investigación es el uso de los diferentes diagramas para representar los requerimientos e interacción sobre el sistema, así como la comparación detallada por parámetros aplicada a los *framework* Ruby On Rails y Django.

Peña y Cambisaca (2015) realizaron la investigación *Análisis comparativo entre los frameworks JavaScript MVC, AngularJS y Ember.Js para el desarrollo de aplicaciones Web. Caso práctico: Sistema de control de botiquín veterinario para el Ministerio de Agricultura, Ganadería, Acuacultura y Pesca - MAGAP, Morona Santiago*, desarrollado para obtener el título profesional de Ingeniero en Sistemas Informáticos en la Escuela Superior Politécnica de Chimborazo - Ecuador, teniendo como objetivo realizar un análisis comparativo entre los frameworks JavaScript Modelo-Vista-Controlador AngularJS y Ember.Js, para luego el sistema de control de botiquín veterinario para el Ministerio de Agricultura, Ganadería, Acuacultura y Pesca, utilizando la metodología *Rational Unified Process* (RUP), llegando a la conclusión que los indicadores especificados para medir el rendimiento permitieron obtener como resultado, que AngularJS alcanzó un 48,89%, mientras que EmberJS obtuvo un 30,83%, dando como resultado final que el *framework* AngularJS es el más adecuado para aplicaciones Web que requieran de un mejor rendimiento y recomendando que al realizar un análisis comparativo entre *framework* de desarrollo, se debe elegir cuidadosamente los *frameworks* involucrados, los cuales deben tener funcionalidad y características semejantes para obtener resultados certeros sin inconvenientes. El aporte de la investigación es la comparación de los distintos parámetros que se deben tener en cuenta al indagar cómo es el rendimiento de un determinado *framework*.

Suárez (2011) realizó la investigación *Análisis comparativo de los frameworks EJB3 y ADO.NET Entity Framework para el desarrollo de aplicaciones empresariales*, desarrollado para obtener el título profesional de Ingeniero en Sistemas Informáticos y de Computación en la Escuela Politécnica Nacional de Quito - Ecuador, teniendo como objetivo comparar los *frameworks Enterprise JavaBeans* 3 y ADO.NET *Entity Framework* para determinar cuál *framework* es más idóneo para desarrollar aplicaciones empresariales, utilizando la metodología ICONIX, llegando a la conclusión de que no existe una diferencia marcada entre los *frameworks* evaluados, por lo que ambos serían aptos para la creación de una aplicación empresarial, las características técnicas en los ambientes de cada *framework* son similares, por lo cual la decisión en una organización para optar por uno u otro queda a decisión y análisis de criterios administrativos para lo cual puede aplicar la misma metodología de comparación; recomendando que se debería considerar el análisis de *frameworks* adicionales para aplicaciones empresariales, como el caso de Hibernate y Spring que cuentan con versiones para Java y .Net y que durante el periodo de desarrollo de este trabajo tuvieron notables avances y aceptación en el mercado. El aporte de la investigación es la metodología usada para realizar un análisis comparativo, definiendo el ámbito, criterios y exclusiones.

Spano (2009) realizó la investigación *Usando frameworks Web para el desarrollo de sistemas de información*, para obtener el grado de magister en Ciencias de la Computación en la Universidad Carolina – República Checa, teniendo como objetivo identificar los criterios de comparación adecuados para *framework* Web y comparar los *frameworks* Spring, Struts 2 y JBoss Seam, llegando a la conclusión de que Spring posee el mejor tiempo de respuesta en las interfaces de usuario seguido de Struts, JBoss Seam fue el más lento de los tres *frameworks* pero presenta una curva de aprendizaje menor que el resto de *frameworks* comparados; recomendando que se debería considerar el utilizar Spring para proyectos de larga escala pues ofrece integración con distintas tecnologías tanto en el lado servidor como en el lado cliente, Struts 2 para proyectos a pequeña escala pues en proyectos a gran escala tendería a incrementar los tiempos de desarrollo, JBoss Seam para proyectos en los cuales la *performance* de la aplicación final no es crucial. El aporte de la investigación son los datos respecto al *performance* de los *frameworks* Spring y Struts 2 en versiones anteriores, lo que permite realizar una comparación respecto a la evolución de ambos *frameworks*.

2.2 Bases teóricas

La presente investigación se realizó en base a las aplicaciones Web, específicamente en los *frameworks* Web Java más populares en la actualidad, los cuales están basados en distintos estándares y patrones de diseño. Previo a la definición de los *frameworks* mencionados, es importante definir que es un sitio y aplicación Web.

2.2.1 Sitio Web y aplicación Web

Los sitios Webs son "un conjunto de páginas Web interconectadas, que generalmente incluyen una página principal, generalmente ubicadas en el mismo servidor, preparadas y mantenidas como una recopilación de información por parte de una persona, grupo u organización" (The American Heritage® Science Dictionary, s.f.). Por el contrario, "una aplicación Web es un sitio Web que contiene páginas con contenido sin determinar, parcialmente o en su totalidad. El contenido final de una página se determina sólo cuando el usuario solicita una página del servidor Web" (Adobe Systems Incorporated, 2017). El desarrollo de las aplicaciones Web hoy en día se lleva a cabo mediante la utilización de variadas herramientas, como por ejemplo *frameworks*, los cuales simplifican las tareas de desarrollo y son explicados a continuación.

2.2.2 Frameworks Web

> Los *frameworks* Web del lado del servidor (también conocidos como "*frameworks* de aplicaciones Web") son *frameworks* de *software* que hacen que sea más fácil escribir, mantener y escalar aplicaciones Web. Proporcionan herramientas y bibliotecas que simplifican las tareas de desarrollo Web comunes, incluyendo el enrutamiento de las URL a apropiados manejadores, la interacción con bases de datos, soportando sesiones y autorización de usuarios, formatos de salida (por ejemplo, HTML, JSON, XML) y mejorando la seguridad contra ataques Web (Mills y Willee, 2017).

Los *frameworks* Web se han convertido en una herramienta bastante utilizada a lo largo del desarrollo de aplicaciones, en especial, en el desarrollo de aplicaciones Web, donde las tareas de mantenimiento o escalabilidad pueden llegar a tener un alto grado de complejidad. Todo ello siguiendo una determinada arquitectura, la cual es detallada a continuación.

2.2.3 Arquitectura cliente/servidor

Las aplicaciones Web son desarrolladas bajo la arquitectura cliente servidor, en este sentido, según Janssen y Janssen, Client/Server Architecture (s.f.):

> La arquitectura de cliente / servidor es un modelo informático en el que el servidor aloja, entrega y administra la mayoría de los recursos y servicios que consumirá el cliente. Este tipo de arquitectura tiene una o más computadoras cliente conectadas a un servidor central a través de una red o conexión a Internet.
>
> De esta definición, se pueden reconocer dos partes, el *front-end* y el *back-end*.

"El *front-end* de una aplicación es claramente humana. Es lo que el usuario ve y experimenta. El *front-end* de una aplicación es menos sobre el código y más acerca de cómo un usuario interpretará la interfaz en una experiencia" (Bloc, 2016).

"El *back-end* de una aplicación es responsable de la lógica de negocio, interacciones con las bases de datos y el rendimiento. La mayor parte del código que se requiere para hacer que una aplicación funcione se hará en el *back-end*" (Bloc, 2016).

Hoy en día existe una gran variedad de lenguajes de programación orientados al desarrollo Web, pero originalmente todos ellos empezaron basados en aplicaciones de escritorio, según un estudio realizado por TIOBE (2018), Java se ubica en el primer lugar con 16.881% de popularidad. A ello TIOBE coloca a Java, Python y Visual Basic .NET como los tres principales lenguajes MVC más populares (**ver ANEXO N° 01 y ANEXO N° 02**).

Visto el dominio que posee Java como lenguaje de programación, es común esperar una amplia lista de frameworks disponibles para este lenguaje, según ZeroTurnaround (2017), existen varios *frameworks* Web que lideran el desarrollo de aplicaciones Web en Java, como se puede apreciar en la siguiente figura:

Rank	Framework	Popularity
1	Spring mvc	29.39
2	JSF	15.19
3	Spring Boot	11.69
4	GWT	7.6
5	Grails	6.73
6	Struts	7.47
7	Play framework	4.16
8	Seam	1.88
9	jax-rs	3.1
10	Vaadin	2.45
11	Wicket	1.92
12	Tapestry	1.83
13	JHipster	0.73
14	Dropwizard	1.05
15	Sparkjava	0.76
16	Lagom	0.71
17	Vert.x	0.72
18	Ratpack	0.15
19	Rapidoid	0

Figura 2.1. Ranking de popularidad de *frameworks* Web para Java *Enterprise Edition*.
Fuente: Java Web *Frameworks Index* (ZeroTurnaround, 2017)

Dentro del *ranking* mostrado en la figura anterior, se puede observar el dominio completo que tiene Spring Framework, cabe resaltar que se mencionan *frameworks* que usan como base a Java, pero las aplicaciones Web no se programan en dicho lenguaje (**ver ANEXO N° 03**). De ello, Spring Framework y Struts son *frameworks back-end* que poseen el mayor grado de popularidad, y permiten el desarrollo de aplicaciones Web en el lenguaje Java. JavaServer Faces (JSF) no ha sido tomado en consideración, pese a ocupar el segundo lugar puesto que, no es un *framework back-end*, es un *framework front-end* tal y como lo indica Oracle Corporation (s.f): "La tecnología JavaServer Faces establece el estándar para construir interfaces de usuario del lado del servidor".

2.2.4 Spring *framework*

Es uno de los *frameworks* de código abierto más populares en el mercado actualmente, se basa en el patrón modelo – vista – controlador (MVC), agilizando el desarrollo de aplicaciones y provee de herramientas que permiten la integración con otras plataformas.

> Spring Framework proporciona un modelo de programación y configuración integral para aplicaciones empresariales modernas basadas en Java, en cualquier tipo de plataforma de implementación. Un elemento clave de Spring es el soporte de infraestructura a nivel de aplicación: Spring se enfoca en la "fontanería" de aplicaciones empresariales para que los equipos puedan enfocarse en la lógica empresarial a nivel de aplicación, sin lazos innecesarios con entornos de despliegue específicos (Spring, 2017).

Spring es un *framework* modular, lo cual permite elegir qué partes van a ser usadas en el desarrollo de aplicaciones, y cuáles pueden ser agregadas en un futuro o dejadas de lado, todo ello, sin complicar el ciclo de desarrollo, los módulos que posee se muestran en la siguiente figura:

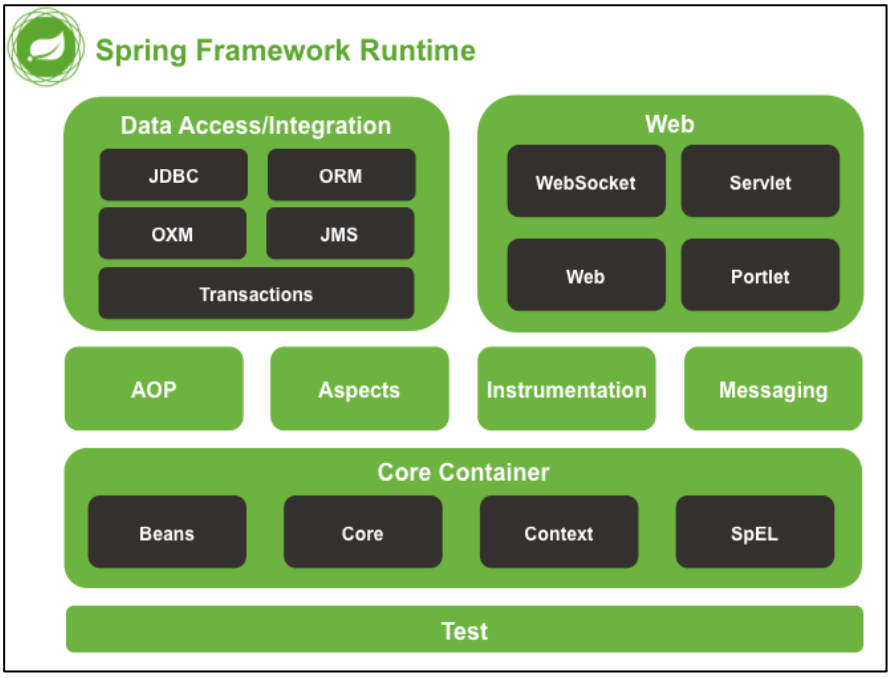


Figura 2.2. Módulos que comprenden Spring *Framework*.
Fuente: Spring *Framework* (Spring, 2017)

El modulo Test "contiene el *Framework Test* que soporta las pruebas sobre los componentes de Spring usando JUnit o TestNG" (Spring, 2017). El módulo *Core*, posee las herramientas necesarias para el trabajo con los Java *Beans* y su distribución a las capas superiores de las aplicaciones. El módulo AOP "proporciona una implementación de programación orientada a aspectos conforme a Alianza AOP" (Spring, 2017). El módulo *Aspects* "provee soporte para la integración con AspectJ" (Spring, 2017). El módulo *Instrumentation* "proporciona soporte de instrumentación de clase y de cargador de clase para ser utilizado en ciertos servidores de aplicaciones" (Spring, 2017). El módulo *Messaging* "provee un amplio soporte para la integración con sistemas de mensajería" (Spring, 2017). El modulo Web "proporciona funciones de integración básicas orientadas a la Web, como la funcionalidad de carga de archivos de varias partes, la inicialización del contenedor IoC. También contiene las partes relacionadas con la Web del soporte remoto de Spring" (Spring, 2017). El módulo *Data Access/Integration* se encarga

de proveer abstracción para la operación con los datos, permitiendo la integración con distintos frameworks de acceso a datos y de transacciones.

Spring cuenta con una configuración basada en anotaciones y una configuración basada en archivos XML, la configuración basada en anotaciones fue introducida en la versión 2.5 con un soporte mínimo posteriormente fue adicionado el soporte completo para realizar configuración de los metadatos mediante anotaciones en la versión 3.0. La configuración del Spring consiste en al menos una definición, y típicamente más de una definición de *beans* que el contenedor debe administrar. Los metadatos de configuración basados en XML muestran estos *beans* configurados como elementos <*bean* /> dentro de un elemento de nivel superior <*beans* />. La configuración de Java generalmente usa métodos anotados *@Bean* dentro de una clase *@Configuration*. Estas definiciones de *bean* corresponden a los objetos reales que componen la aplicación. Normalmente se definen objetos de capa de servicio, objetos de acceso a datos (DAO), objetos de presentación como instancias de los *Action* de Struts, objetos de infraestructura como Hibernate *SessionFactories*, JMS *Queues*, etc.

La introducción de configuraciones basadas en anotaciones planteó la pregunta de si este enfoque es "mejor" que XML. La respuesta corta es que depende. La respuesta larga es que cada enfoque tiene sus pros y sus contras, y generalmente le corresponde al desarrollador decidir qué estrategia conviene más. Debido a la forma en que se definen, las anotaciones proporcionan mucho contexto en su declaración, lo que lleva a una configuración más breve y concisa. Sin embargo, XML se destaca en el enlace de componentes sin tocar su código fuente o recompilarlos. “Algunos desarrolladores prefieren tener el enlazado cerca del código fuente, mientras que otros argumentan que las clases anotadas ya no son POJO y, además, que la configuración se vuelve descentralizada y más difícil de controlar” (Spring, 2017). Spring puede acomodar ambos estilos e incluso mezclarlos. A través de su opción *JavaConfig*, Spring permite que las anotaciones se utilicen de forma no invasiva, sin tocar el código fuente de los componentes de destino.

En la configuración basada en anotaciones, Spring proporciona más anotaciones estereotipadas: *@Component*, *@Service* y *@Controller*. *@Component* es un estereotipo genérico para cualquier componente gestionado por Spring. “*@Repository*, *@Service* y *@Controller* son especializaciones de @Component para casos de uso más específicos, por ejemplo, en las capas de persistencia, servicio y presentación, respectivamente” (Spring, 2017). Por lo tanto, se pueden anotar las clases de los componentes con *@Component*, pero al anotarlas con *@Repository*, *@Service* o *@Controller*, las clases son más adecuadas para el procesamiento por herramientas o la asociación con aspectos. Por ejemplo, estas anotaciones de estereotipo son objetivos ideales para los puntos de corte o referencia. “Es posible que *@Repository*, *@Service* y *@Controller* puedan tener una semántica adicional en futuras versiones de Spring” (Spring, 2017). Por lo tanto, si se debe elegir entre usar *@Component* o *@Service* para la capa de servicio, *@Service* es claramente la mejor opción.

> De forma predeterminada, las clases anotadas con *@Component*, *@Repository*, *@Service*, *@Controller* o una anotación personalizada que se anota con *@Component* son los únicos componentes candidatos detectados. Sin embargo, se puede modificar y ampliar este comportamiento simplemente aplicando filtros personalizados. Añadiéndolos como parámetros *includeFilters* o *excludeFilters* de la anotación *@ComponentScan* (o como subelementos *include-filter* o *exclude-filter* del elemento *component-scan*). Cada elemento de filtro requiere los atributos de tipo y expresión (Spring, 2017).

Dentro del contenedor en sí, las definiciones de los *beans* se representan como objetos *BeanDefinition*, que contienen (entre otra información) los siguientes metadatos:

- Un nombre de clase calificado para el paquete: típicamente la clase de implementación real del *bean* que se está definiendo.
- Elementos de configuración de comportamiento de los *beans*, que establecen cómo debe comportarse el *bean* en el contenedor (alcance, devoluciones de llamadas del ciclo de vida, etc.).
- Las referencias a otros *beans* que se necesitan para que el *bean* haga su trabajo; estas referencias también se llaman colaboradores o dependencias.
- Otras configuraciones para establecer en el objeto recién creado, por ejemplo, la cantidad de conexiones que se usarán en un *bean* que administra un grupo de conexiones, o el límite de tamaño del grupo.

Cada *bean* tiene uno o más identificadores. Estos identificadores deben ser únicos dentro del contenedor que aloja al *bean*. Un *bean* generalmente solo tiene un identificador, pero si se requiere más de uno, los adicionales pueden considerarse alias. En los metadatos de configuración basados en XML, se utilizan los atributos id y/o *name* para especificar los identificadores del *bean*. El atributo *id* le permite especificar exactamente una identificación. Convencionalmente, estos nombres son alfanuméricos (*'myBean'*, *'fooService'*, etc.), pero también pueden contener caracteres especiales. Si se desea introducir otros alias para el *bean*, también se pueden especificar en el atributo de nombre, separados por una coma (,), punto y coma (;) o espacio en blanco. "En versiones anteriores a Spring 3.1, el atributo *id* se definió como un tipo xsd: ID, que restringía los posibles caracteres. A partir de 3.1, se define como un tipo xsd: *string*" (Spring, 2017). Se debe tener en cuenta que la singularidad del *id* del *bean* sigue siendo impuesta por el contenedor, aunque ya no por los analizadores XML.

No es necesario que se proporcione un nombre o una identificación para un *bean*. Si no se proporciona un nombre o una identificación explícitamente, el contenedor genera un nombre único para ese *bean*, "sin embargo, si se desea hacer referencia a ese *bean* por su nombre, mediante el uso del elemento *ref* o la búsqueda de estilo de Localizador de Servicios, se debe proporcionar un nombre" (Spring, 2017). Las motivaciones para no proporcionar un nombre están relacionadas con el uso de *beans* internos y colaboradores de auto envío.

El modelo de transacciones usado en Spring es una abstracción del modelo estándar de transacciones usado por los distintos entornos transaccionales en Java EE. "La clave para la abstracción de transacciones de Spring es la noción de una estrategia de transacción. Una estrategia de transacción se define mediante la interfaz *org.springframework.transaction.PlatformTransactionManager*" (Spring, 2017).

```
public interface PlatformTransactionManager {

    TransactionStatus getTransaction(TransactionDefinition definition) throws TransactionException;

    void commit(TransactionStatus status) throws TransactionException;

    void rollback(TransactionStatus status) throws TransactionException;
}
```

Figura 2.3. Interface *PlatformTransactionManager*.
Fuente: Abstracción del modelo de transacciones (Spring, 2017)

La interface de la Figura 2.3 es principalmente una interfaz de proveedor de servicio (SPI), aunque se puede utilizar mediante programación desde el código de la aplicación. Debido a que *PlatformTransactionManager* es una interfaz, se puede burlar o moldear fácilmente según sea necesario. No está vinculado a una estrategia de búsqueda como JNDI. Las implementaciones *PlatformTransactionManager* se definen como cualquier otro objeto (o *bean*) en el contenedor IoC. Este beneficio solo hace que las transacciones de Spring *Framework* sean una abstracción que vale la pena incluso cuando se trabaja con JTA. El código transaccional puede probarse mucho más fácilmente que si se usara JTA directamente.

De acuerdo con la filosofía de Spring, "las excepciones *TransactionException* que pueden arrojarse por cualquiera de los métodos de la interfaz *PlatformTransactionManager* son en tiempo de ejecución (es decir, extienden la clase *java.lang.RuntimeException*)" (Spring, 2017). Las fallas de la infraestructura de la transacción son casi invariablemente fatales. En casos excepcionales en los que el código de la aplicación puede recuperarse de una falla de la transacción, el desarrollador de la aplicación aún puede elegir capturar y manejar la excepción *TransactionException*. El punto sobresaliente es que los desarrolladores no están obligados a hacerlo.

El método *getTransaction* (..) devuelve un objeto *TransactionStatus*, según un parámetro de *TransactionDefinition*. El *TransactionStatus* devuelto puede representar una nueva transacción, o puede representar una transacción existente si existe una transacción coincidente en la pila de llamadas actual. "La implicación en este último caso es que, al igual que con los contextos de transacción Java EE, un estado de transacción está asociado con un hilo de ejecución" (Spring, 2017).

La interfaz *TransactionDefinition* especifica:

- Propagación: Por lo general, todo el código ejecutado dentro del alcance de una transacción se ejecutará en esa transacción. Sin embargo, se tiene la opción de especificar el comportamiento en el caso de que se ejecute un método transaccional cuando ya existe un contexto de transacción. Por ejemplo, el código puede continuar ejecutándose en la transacción existente (el caso común); o la transacción existente puede suspenderse y crearse una nueva transacción. Spring ofrece todas las opciones de propagación de transacciones familiares a EJB CMT.

- Aislamiento: El grado en que esta transacción está aislada del trabajo de otras transacciones. Por ejemplo, ¿esta transacción puede ver escrituras no confirmadas de otras transacciones?

- Tiempo de espera: Durante cuánto tiempo se ejecuta esta transacción antes de que se agote el tiempo de espera y la infraestructura de transacciones subyacente la retrotraiga automáticamente.

- Estado de solo lectura: se puede usar una transacción de solo lectura cuando su código lee pero no modifica los datos. Las transacciones de solo lectura pueden ser una optimización útil en algunos casos, como cuando se está usando Hibernate. Estas configuraciones reflejan conceptos transaccionales estándar.

La interfaz *TransactionStatus*, mostrada en la Figura 2.4, proporciona una forma simple para que el código transaccional controle la ejecución de la transacción y el estado de la transacción de consulta.

```
public interface TransactionStatus extends SavepointManager {

    boolean isNewTransaction();

    boolean hasSavepoint();

    void setRollbackOnly();

    boolean isRollbackOnly();

    void flush();

    boolean isCompleted();

}
```

Figura 2.4. Interface *TransactionStatus*.
Fuente: Interface *TransactionStatus* (Spring, 2017)

Independientemente de si se opta por la gestión declarativa o programática de transacciones en Spring, la definición correcta de la implementación *PlatformTransactionManager* es absolutamente esencial. Por lo general, se define esta implementación a través de la inyección de dependencias. “Las implementaciones de *PlatformTransactionManager* normalmente requieren conocimiento del entorno en el que trabajan: JDBC, JTA, Hibernate, etc” (Spring, 2017).

2.2.5 Struts 2 *framework*

"Struts 2 es un *framework* para aplicaciones Web basado en el patrón de diseño MVC. Struts 2 no es una nueva versión de Struts 1, sino que, es una reescritura completa de la arquitectura Struts" (Apache Software Foundation, Inc, 2015). El *framework* WebWork inicialmente comenzó con el *framework* Struts como su base y su objetivo era ofrecer un *framework* mejorado basado en Struts para facilitar el desarrollo Web para los desarrolladores. Después de un tiempo, el *framework* WebWork y la comunidad Struts se unieron para crear Struts 2.

Para facilitar la presentación de datos dinámicos, Struts incluye una biblioteca de etiquetas de marcado. Las etiquetas interactúan con las características de validación e internacionalización de Struts, para garantizar que la entrada sea correcta y la salida esté localizada. La biblioteca de etiquetas se puede usar con JSP o con sistemas de plantillas como *FreeMarker* o *Velocity*.

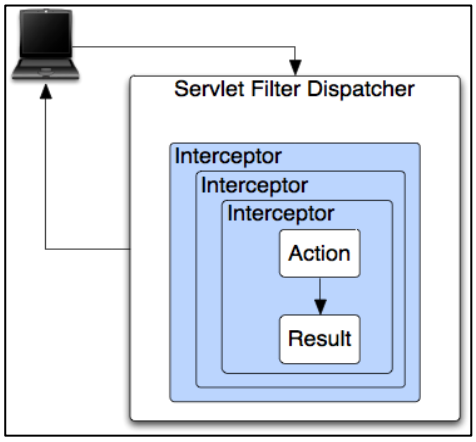


Figura 2.5. Procesamiento de peticiones.
Fuente: Apache Struts 2 (Apache Software Foundation, Inc, 2015)

De la figura anterior, se toman las etapas que atraviesa una petición dentro de una aplicación Web desarrollada con el *framework* Struts:

- El navegador Web solicita un recurso (/*mypage.action*, */reports/myreport*.pdf, etcétera)
- *Filter Dispatcher* examina la solicitud y determina la acción apropiada
- Los Interceptores aplican automáticamente una funcionalidad común a la solicitud, como flujo de trabajo, validación y manejo de carga de archivos
- El método de acción se ejecuta, generalmente almacenando y / o recuperando información de una base de datos.
- *Result* representa el resultado que será enviado al navegador, ya sea HTML, imágenes, PDF u otro elemento.

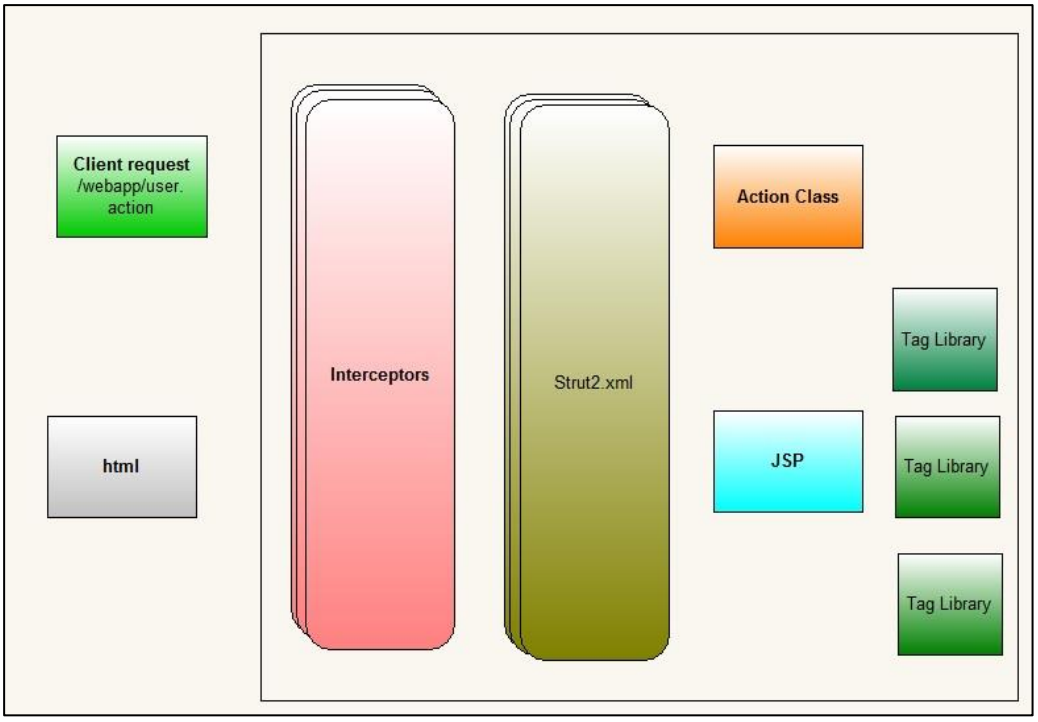


Figura 2.6. Arquitectura de Struts 2.
Fuente: Apache Struts 2 (Apache Software Foundation, Inc, 2015)

Interceptor o Interceptores pueden ejecutar código antes y después de invocar un *Action*, "la mayor parte de la funcionalidad principal del *framework* se implementa como interceptores. Todos los interceptores se pueden conectar, por lo que puede decidir exactamente qué funciones necesita un *Action* soportar." (Mandliya, 2012). El archivo Strut2.xml "actúa como un enrutador el cual invoca las clases *Action* para las peticiones del cliente" (Mandliya, 2012). *Action* es "considerado como una clase modelo la cual invoca la lógica de función" (Mandliya, 2012). JSP "es la parte de la vista, dependiendo del *Action*, el correspondiente archivo JSP es renderizado y el resultado es retornado al usuario" (Mandliya, 2012). *Tag Library* son un conjunto de librerías que permiten la manipulación de los datos en los archivos JSP.

Struts se presenta como un *core* o un solo núcleo, a diferencia de Spring que posee módulos especializados para cada elemento de configuración presente y en uso en la aplicación que se desarrolle. En contraparte a los módulos de Spring, Struts posee un amplio repertorio de *plugins*, que mejoran, expanden o agregan funcionalidades a su *core*, entre algunos de ellos se encuentran:

- *Plugin* de validación de *beans*: *plugin* que permite realizar validaciones sobre cada una de las propiedades presentes en un *bean*.
- *Plugin* CDI: encargado de la gestión del contenedor de inyección de dependencias, este *plugin* realiza la configuración necesaria para que las aplicaciones desarrolladas con Struts puedan comunicarse y hacer uso de la mayoría de contenedores de inyección de dependencias presentes en el mercado, teniendo como base la especificación Java JSR 299, que define los contextos y la inyección de dependencias para la plataforma de Java EE.
- *Plugin* de convenciones: provee una variedad de configuraciones predefinidas que pueden ser usadas por los desarrolladores para agilizar los procesos de construcción de aplicaciones, incluyendo herramientas para las operaciones con anotaciones que son agregadas por otro *plugin*.

- *Plugin* de anotaciones: este *plugin* permite reemplazar todos los archivos de configuración XML usados con Struts, por una configuración basada en las anotaciones dadas con Java 5.
- *Plugin* de configuración del navegador: permite enlazar la aplicación a un monitor definido dentro del *plugin* y que permite ver el estado de la aplicación, así como de todos los componentes que posee.
- *Plugin* de soporte para Java8: *plugin* que agrega soporte para las instrucciones y nuevas funciones como las expresiones lambda.
- *Plugin* para JSF: *plugin* que permite la integración con *JavaServerFaces*.
- *Plugin* para JSON: permite el envío y recepción de datos mediante el formato JSON.
- *Plugin* para JUnit: permite la integración con el *framework* de testeo JUnit, facilitando la operación con los *Actions*.

Al igual que el *framework* Spring, Struts 2 posee su propio contenedor de inyección de dependencias, el cual es similar a *Google Guice*. "Los complementos están disponibles para integrar aplicaciones con otros contenedores de inyección de dependencias (por ejemplo, Spring *Plugin*, *Plexus Plugin*). Una aplicación incluso puede usar una copia local de *Google Guice* para las necesidades de inyección de dependencia" (Apache Software Foundation, Inc, 2015). Tanto la Inyección de Dependencias como *Inversion of Control* (IoC) se explica a continuación.

2.2.6 *Inversion of control (IoC) y dependency injection (DI)*

El término *inversion of control (IoC)* se refiere a un estilo de programación en el que un *framework* o *runtime*, controla el flujo del programa. La inversión del control "significa que estamos cambiando el control de la manera normal" (Chauhan, 2013). Mientras que "el patrón de inyección de dependencias utiliza un objeto generador para inicializar los objetos y proporciona las dependencias necesarias para el objeto, lo que significa que le permite inyectar una dependencia desde fuera de la clase" (Chauhan, 2013).

Un ejemplo de Inversión de control es mostrado en la siguiente imagen, en la cual se modela la forma en que trabaja una aplicación de chat.

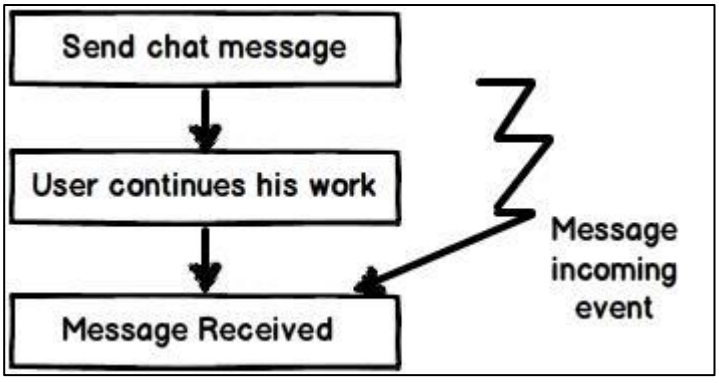


Figura 2.7. Aplicación de chat.
Fuente: *Dependency Injection (DI) vs. Inversion of Control (IOC)* (Koirala, 2014)

En la imagen anterior se puede observar que el control de las actividades no está en control del programa, sino de un ente externo, en este caso el evento que se genera con la llegada de un nuevo mensaje, maneja el flujo. Cuando un mensaje llega, se genera una respuesta, sin que el programa mismo la genere en un orden establecido. Esto es un claro ejemplo de concepto de inversión de control, pero este concepto no solo es implementado por eventos, sino que se puede delegar el control utilizando cualquier otra forma, por ejemplo, delegando el control a contenedores, patrones observador, etc.

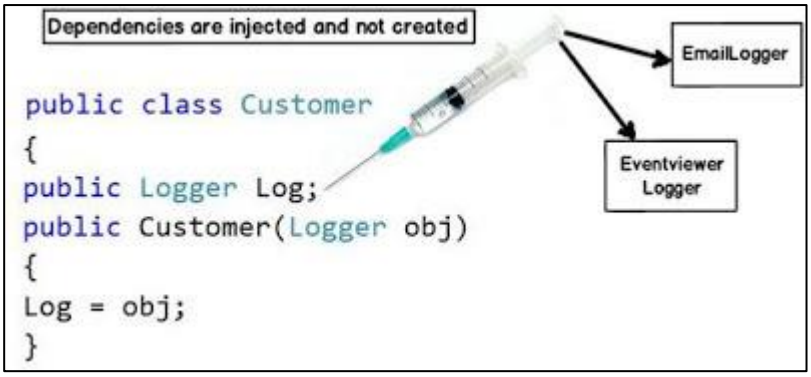


Figura 2.8. Ejemplo de Inyección de dependencias.
Fuente: *Dependency Injection (DI) vs. Inversion of Control (IOC)* (Koirala, 2014)

En la imagen anterior se muestra la forma en la que inyección de dependencias trabaja, se puede observar que la variable *Log* de tipo *Logger* será "inyectada" ya sea, con una instancia de *EmailLogger* o *EventviewerLogger*, todo ello sin llegar a instanciar en el código de la clase *Customer* el objeto, en este caso, la inyección de la dependencia se realiza a través del constructor.

"La principal meta de la inversión de control y la inyección de dependencias es de remover las dependencias de la aplicación. Esto hace al sistema más desacoplado y mantenible" (Koirala, 2014).

Tanto Spring como Struts 2 hacen uso de inyección de dependencias, esto puede ser usado para extender aún más las posibilidades de uso de cada *framework*, simplificando el desarrollo, así mismo, permite que sean empleados en conjunto con otros *frameworks*, entre ellos, los encargados del manejo de los datos, conocidos como ORMs los cuales se explican a continuación.

2.2.7 *Object-relational mapping* (ORM)

"Mapeo objeto-relacional convierte los datos entre sistemas de tipos que no pueden coexistir dentro de bases de datos relacionales y lenguajes orientados a objetos" (Janssen y Janssen, s.f.). Al usar un ORM, se usa el término entidades, que son clases que hacen referencia a tablas en la base de datos, la instancia de las clases se refiere a las filas y los atributos de las instancias de las clases se refieren a las columnas de la tabla. Esto proporciona soluciones a los problemas que surgen al desarrollar aplicaciones de persistencia utilizando el método JDBC tradicional. Esto también reduce el código que debe escribirse. En el lenguaje de programación Java, existen diferentes *frameworks* que implementen esta técnica de programación, de ellos destaca Hibernate, el cual fue usado en la presente investigación y se explica a continuación.

2.2.8 Hibernate

Es una herramienta de mapeo objeto-relacional (ORM) desarrollada para la plataforma Java, la cual facilita el mapeo de atributos entre una base de datos relacional tradicional y los objetos de una aplicación, esto se puede realizar mediante archivos declarativos (XML) o anotaciones en los *beans* de las entidades que se usarán para el mapeo de la base de datos en el modelo de clases.

Hibernate "es una implementación de la interface de programación de aplicaciones (API) de Java, llamada Java *Persistence* API (JPA)" (Jboss, Inc, s.f). Ello conlleva a poder ser usada en cualquier ambiente que soporte JPA incluyendo la versión de escritorio de Java (Java SE).

> Permite desarrollar clases persistentes siguiendo expresiones idiomáticas orientadas a objetos naturales que incluyen herencia, polimorfismo, asociación, composición y el *framework* de colecciones de Java. Hibernate no requiere interfaces o clases base para clases persistentes y permite que cualquier clase o estructura de datos sea persistente (Jboss, Inc, s.f).

Hibernate, como cualquier otra herramienta usada en el desarrollo de aplicaciones, debe ser usado de la forma correcta, esto se traduce en el uso correcto de cada uno de sus componentes, ello con el fin de utilizar las mejoras con las que cuenta, entre ellas, está el uso de sus estrategias para la generación de claves principales, mediante el uso de algoritmos que reducen la cantidad de consultas a realizarse al momento de obtener una clave primaria de un registro que ha sido guardado en la base de datos. Otro punto importante respecto a Hibernate es su configuración, de forma predeterminada, Hibernate hace uso de las cargas perezosas o *Lazy Loads*, esto con el fin de evitar las consultas innecesarias a la base de datos, ahorrando tiempo y recursos, esta forma de trabajo suele ser la indicada en la mayoría de casos, para casos en los que se necesiten todos los datos al momento de realizar la consulta, existen la cargas ansiosas o *Eager Loads*, el inconveniente que acarrea este tipo de cargas, es el hecho de obtener todos los datos relacionados con la entidad en uso, lo que por ejemplo, representa un problema si se trabajasen con una cantidad inmensa de datos, debido a que Hibernate traerá, por ejemplo, la lista de todos los elementos que se encuentren en una relación *OneToMany* o *ManyToMany*, generando problemas de sobrecarga en el sistema y por consiguiente, elevando los tiempos de espera.

Para el desarrollo de la respectiva aplicación Web con los *frameworks* indicados anteriormente, se utilizó una metodología que responda a las necesidades del proyecto de investigación, para la cual se eligió la siguiente:

2.2.9 *Rational unified process* (RUP)

Es un marco que describe el proceso de desarrollo de software, el cual fue desarrollado por la empresa *Rational Software*, y actualmente es propiedad de *International Business Machines* (IBM), junto con el Lenguaje Unificado de Modelado (UML), constituye una de las metodologías estándar más utilizada para el análisis, diseño, implementación y documentación de sistemas orientados a objetos.

> El Proceso Racional Unificado de IBM (RUP) se trata de un desarrollo de *software* exitoso. Hay tres elementos centrales que definen RUP: un conjunto subyacente de filosofías y principios para el desarrollo de *software* exitoso, un marco de contenido de métodos reutilizables y bloques de creación de procesos, y finalmente el método subyacente y el lenguaje de definición de procesos, el lenguaje de modelado unificado (UML) para el proceso de *software* ingeniería, los idiomas (Péraire, Edwards, Fernandes, Mancin y Carroll, 2007).

La metodología RUP, basa su ciclo de vida en repetir una serie de ciclos que constituyen la vida de un sistema. Cada ciclo concluye con una versión del producto para los clientes, que es una versión entregable o funcional; por lo que es una metodología iterativa e incremental. Cada ciclo consta de cuatro fases: inicio, elaboración, construcción y transición, cada fase se subdivide a la vez en iteraciones. Las iteraciones aportan control al marco de trabajo para descubrir errores con anticipación y poder solucionarlos de la mejor manera, logrando evitar que esto afecte de manera significativa lo planificado.

En la fase de inicio se desarrollará la descripción y requerimientos del sistema y se presenta el análisis del negocio para el producto. Durante la fase de elaboración se especificarán en detalle la mayoría de los casos de uso y se diseñará la arquitectura del sistema. "Establecer el alcance del proyecto de *software* y las condiciones limitantes, incluida una visión operativa, los criterios de aceptación, y lo que se pretende que esté en el producto y lo que no es" (Péraire, Edwards, Fernandes, Mancin y Carroll,

2007). Durante la fase de construcción se creará el producto, en esta fase, se emplean todos los recursos requeridos y planificados. Pero, puede que no está completamente libre de defectos; los cuales se descubrirán y solucionarán durante la fase de transición.

Las fases de desarrollo de la metodología RUP llevan a determinar el tiempo de desarrollo, tiempo que debe ser estimado y tiene vital importancia a lo largo del desarrollo del proyecto.

2.2.10 Tiempo de desarrollo

"Desarrollo es un conjunto de actividades que llevan la formulación de un problema a un programa de computadora bien documentado que resuelve el problema" (Sahni y Kumar, 2004). Es por ello que el tiempo de desarrollo dependerá de las actividades a llevarse a cabo, en el caso de los módulos Web, como indica Smedia (s.f): "es imposible determinar el tiempo de diseño y desarrollo de un sitio Web si no conocemos la dimensión y alcance de su proyecto".

2.2.11 Pruebas de aplicaciones Web

Las pruebas de software forman parte del ciclo de desarrollo y permiten verificar el buen funcionamiento/calidad de la aplicación.

Existen diferentes tipos de pruebas que se utilizan en cualquier desarrollo de sistemas informáticos y por lo tanto también son aptas para las aplicaciones Web. Sin embargo, dada la naturaleza particular de la Web, también existen herramientas que están orientadas a la prueba, tanto de correcto funcionamiento como de carga/rendimiento, de las aplicaciones Web.

En el caso de las aplicaciones Web, además de las pruebas tradicionales, es necesario comprobar el funcionamiento del servidor Web, sistemas gestores de bases de dato (SGBD), red y el conjunto del sistema. Tanto de manera funcional como realizando un test de carga para comprobar la capacidad del mismo.

> Una prueba Web, también llamada prueba Web declarativa, está compuesta por una serie de solicitudes HTTP. Las pruebas Web funcionan en la capa de protocolo emitiendo solicitudes HTTP. Las pruebas Web no ejecutan JavaScript. Sin embargo, puede simular acciones de JavaScript en tiempo de ejecución utilizando complementos de prueba Web, complementos de solicitud de prueba Web, reglas de extracción o pruebas Web codificadas (Microsoft Developer Network, 2007).

a. Pruebas de rendimiento

> Las pruebas de rendimiento Web se graban al examinar la aplicación Web. Las pruebas se incluyen en las pruebas de carga para medir el rendimiento de la aplicación Web con la carga de varios usuarios. Una prueba de rendimiento Web se puede convertir en un script basado en código que se puede editar y personalizar como cualquier otro código fuente. Por ejemplo, puede agregar construcciones de bucle y bifurcaciones (Microsoft Developer Network, 2016).

Para llevar a cabo las pruebas de rendimiento existen diferentes herramientas de *software* que facilitan la tarea de generar, ejecutar y recolectar la información obtenida luego de la puesta en marcha de una o varias pruebas.

b. Tiempo de respuesta

El tiempo de respuesta es uno de los indicadores que se usan para determinar el rendimiento que pueda tener una aplicación frente a una determinada carga de trabajo. El tiempo de respuesta puede ser definido como "el lapso de tiempo entre el fin de una petición en un sistema informático y el inicio de la recepción de la respuesta" (International Business Machines Corporation, 1993).

c. Tiempo de ejecución

El tiempo de ejecución se puede definir como "el tiempo dado por el sistema a una o un conjunto de tareas determinadas y en el cual se incluye la ejecución de los servicios del mismo sistema operativo" (Brukardt, s.f)

2.2.12 Pruebas estadísticas

Las pruebas estadísticas permiten el análisis de los datos obtenidos de las diferentes fuentes con las que se cuentan al momento de la ejecución de una investigación, para el estudio de los datos obtenidos en el desarrollo de la presente investigación se emplearon dos pruebas estadísticas.

a. Análisis de varianza factorial univariante – ANOVA

El análisis de varianza "permite contrastar la hipótesis de igualdad de medias de las poblaciones definidas por los diferentes niveles en que se puede segmentar un factor o variable dependiente" (Vicéns, J., Herrarte, A., y Medina, E, 2005). Cuando se desea estudiar el efecto de más de un factor sobre la variable dependiente, es preciso recurrir a los modelos factoriales de análisis de varianza que permitan estudiar el efecto de diversos factores, tanto de manera individual como conjunta.

Cuando solo se tiene en cuenta un factor, se estudia su efecto sobre la variable dependiente y se especifican diversos contrastes entre los niveles del factor, si el resultado del ANOVA es significativo. Sin embargo, cuando en el estudio intervienen dos factores, hay tres efectos que deben considerarse: los efectos de cada factor por separado sobre la variable dependiente, que se conocen como efectos principales, y el efecto de la interacción de ambos factores sobre la variable dependiente. Si el número de factores fuera tres, los efectos a estudiar serían siete (tres principales, tres interacciones dobles y una interacción triple). Si el número de factores fueran cuatro, los efectos a estudiar serían 15 (cuatro principales, seis interacciones binarias, cuatro interacciones tripes y una interacción cuádruple), y así sucesivamente.

Como ha sido señalado, estos diseños exploran los efectos de cada factor sobre la variable dependiente y los efectos de la interacción. La hipótesis nula para cada factor dice que las medias poblacionales definidas por los grupos o niveles del factor son iguales. Por su parte, las hipótesis referidas a las interacciones afirman que éstas no existen. Para el contraste de estas hipótesis se utiliza el estadístico F, y según sea su valor crítico se aceptará o no la hipótesis planteada.

El ANOVA requiere el cumplimiento de los siguientes supuestos:

- Las poblaciones (distribuciones de probabilidad de la variable dependiente correspondiente a cada factor) son normales.
- Las K muestras sobre las que se aplican los tratamientos son independientes.
- Las poblaciones tienen todas igual varianza (homocedasticidad).

El ANOVA se basa en la descomposición de la variación total de los datos con respecto a la media global (SCT):

- Variación dentro de las muestras (SCD) o Intra-grupos, cuantifica la dispersión de los valores de cada muestra con respecto a sus correspondientes medias.
- Variación entre muestras (SCE) o Inter-grupos, cuantifica la dispersión de las medias de las muestras con respecto a la media global.

Las expresiones para el cálculo de los elementos que intervienen en el ANOVA son las siguientes:

Media Global: $\overline{X} = \frac{\sum_{j=1}^{k} \sum_{i=1}^{nj} X_{ij}}{n}$

Variación Total: $SCT = \sum_{j=1}^{K} \sum_{i=1}^{nj} (x_{ij} - \bar{X})^2$

Variación Intra-grupos: $SCD = \sum_{j=1}^{K} \sum_{i=1}^{nj} (x_{ij} - \overline{X_j})^2$

Variación Inter-grupos: $SCE = \sum_{j=1}^{K} \sum_{i=1}^{nj} (\overline{X_j} - \bar{X})^2 n_j$

Siendo x_{ij} el i-ésimo valor de la muestra j-ésima; n_j el tamaño de dicha muestra y $\overline{X_j}$ su media

Cuando la hipótesis nula es cierta SCE/K-1 y SCD/n-K son dos estimadores insesgados de la varianza poblacional y el cociente entre ambos se distribuye según una F de Snedecor con K-1 grados de libertad en el numerador y N-K grados de libertad en el denominador. Por lo tanto, si H0 es cierta es de esperar que el cociente entre ambas estimaciones será aproximadamente igual a 1, de forma que se rechazará H0 si dicho cociente difiere significativamente de 1.

b. Diferencia de medias estadísticas para la contrastación de hipótesis

Previo a la definición de la prueba estadística para la diferencia de medias, se deben definir conceptos empleados en párrafos anteriores como lo son hipótesis nula, hipótesis alternativa, errores tipo I y II, nivel de significación, pruebas bilaterales y unilaterales.

La hipótesis nula, "es un supuesto que se pondrá a prueba, por lo cual debe formularse de tal manera que pueda ser aprobada o rechazada" (Minitab Inc, 2017), si la hipótesis nula es aceptada, entonces la hipótesis alternativa debe ser rechaza y viceversa, se simboliza como H0.

La hipótesis alternativa "es un supuesto que contempla todos o un grupo de los supuestos dejados de lado al momento de plantear la hipótesis nula y representa la proposición que el analista o investigador espera aprobar" (Minitab Inc, 2017).

El error tipo I "es aquel error que se puede cometer al rechazar la hipótesis nula cuando realmente ella es verdadera y se simboliza por alfa (α)" (Minitab Inc, 2017). El error tipo II "es aquel error que se puede cometer al aceptar la hipótesis nula cuando realmente ella es falsa y se simboliza por beta (β)" (Minitab Inc, 2017).

El nivel de significación o también llamado nivel de significancia que se simboliza por α, "es la máxima probabilidad que se está dispuesto a asignar al riesgo de cometer un error de tipo I" (Minitab Inc, 2017), con el fin de minimizar el riesgo se utilizan valores bajos como 0.01 o 0.05. Con el fin de evitar influencias en las decisiones que se tomen, el nivel de significancia se asigna antes de realizar la toma de muestras que va a proporcionar el estadístico de la prueba. El estadístico de la prueba "es aquella medida estadística obtenida a través de la muestra, que permite tomar la decisión de aceptar o rechazar la hipótesis nula" (Minitab Inc, 2017). Ante un tamaño de muestra dado, la única forma de disminuir el riesgo de presencia de un error de tipo I, es incrementar el riesgo de presencia de un error de tipo II y viceversa.

Las pruebas bilaterales y unilaterales se emplean en función del planteamiento de la hipótesis nula y alternativa, generalmente la hipótesis nula se plantea como una igualdad, por ejemplo H0: $\mu = 30$, mientras que la hipótesis alternativa se expresa de manera compuesta, por ejemplo H1: $\mu < 30$ o H1: $\mu > 30$ o H1: $\mu \neq 30$.

Si la hipótesis alternativa es del tipo H1: $\mu \neq 30$, la prueba de hipótesis es bilateral o de dos colas, como se muestra en la Figura 2.9.

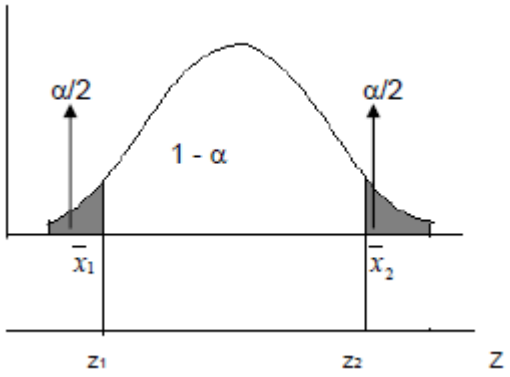


Figura 2.9. Prueba bilateral o de dos colas.

Si es del tipo H1: $\mu < 30$, la prueba de hipótesis es unilateral o de una cola y en este caso a la izquierda, como se muestra en la Figura 2.10.

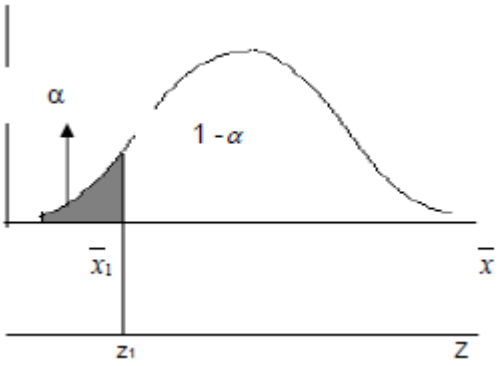


Figura 2.10. Prueba unilateral o de cola izquierda.

Si es del tipo H1: $\mu > 30$ también será una prueba de hipótesis unilateral y en este caso de una cola a la derecha, como se muestra en la Figura 2.11.

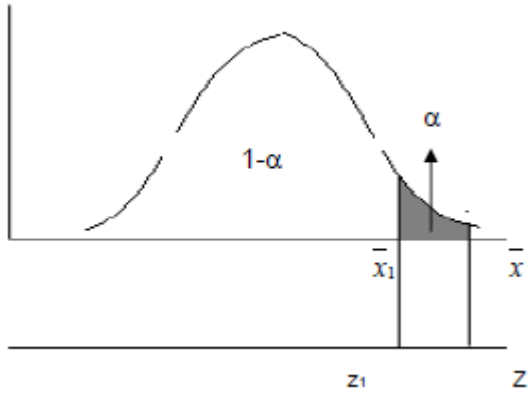


Figura 2.11. Prueba unilateral o de cola derecha.

El área que corresponde a las colas es el nivel de significación α, que "para el caso de una prueba bilateral debe repartirse por iguales partes en ambas colas y que para el caso de una prueba unilateral dicho nivel se concentra en la cola correspondiente" (Canavos, 1992).

Con los conceptos previos explicados, el procedimiento a emplear para la realización de una contrastación de hipótesis mediante la diferencia de medias estadísticas constaría de los pasos siguientes:

- Formular la hipótesis nula y la hipótesis alternativa, se indica que la hipótesis nula de preferencia siempre sea expresada como una igualdad y reflejar el no cambio.
- Especificar el nivel de significación α, teniendo en cuenta los tipos de errores en los que se puede incurrir.
- Formular la regla de decisión, estableciendo para qué valores la hipótesis nula será rechazada.
- Calcular el estadístico sobre el cual se tomará la decisión, el cual es calculado mediante los datos muestrales y debe ser coherente con el parámetro poblacional que se está sometiendo a prueba, es decir, si por ejemplo se somete a prueba una diferencia de medias, el estadístico correspondiente que se deberá calcular será la diferencia de medias muestrales. Se emplea la fórmula $$t = \frac{\overline{x} - \mu_0}{s/\sqrt{n}}$$

  Donde μ_0 es el valor puntual a evaluar, $\bar{x}$ es la media, s la desviación estándar y n el tamaño de la muestra.
- Rechazar o aprobar la hipótesis nula, confrontando el valor del estadístico de prueba obtenido con el estadístico según el criterio de decisión, se procederá a rechazar o aceptar la hipótesis nula.

Existen principios que deben ser tomados en cuenta respecto a la distribución en el muestreo de la media, en el cual se dan tres casos:

- Si una población es normal, "las medias muestrales también se distribuirán normalmente, cualquiera sea el tamaño de la muestra" (Canavos, 1992). No obstante, si no se conoce la desviación estándar poblacional (σ), ésta puede ser reemplazada por la desviación estándar de la muestra (S) si el tamaño de la muestra es mayor que 30.

– Según el teorema central del límite, “si una población no es normal o no se conoce si se cumple o no este comportamiento, las medias muestrales se distribuirán aproximadamente como una distribución normal, si el tamaño de la muestra es mayor que 30” (Canavos, 1992).

– Si “una población es normal o está muy cerca de éste comportamiento y por otra parte no se conoce la desviación estándar poblacional (σ) y además el tamaño de la muestra es menor que 30” (Canavos, 1992), entonces, las medias muestrales se distribuirán de acuerdo a la ley t-student.

2.3 Glosario de términos básicos

- Análisis comparativo: Estudio basado en la comparación de parámetros.
- *Core*: Núcleo de los *frameworks* usados en la presente investigación.
- Desviación típica o estándar: Medida del grado de dispersión de los tiempos de respuesta respecto de la media.
- Hipótesis alternativa: Supuesto que se espera probar.
- Hipótesis nula: Afirmación inicial basada en un análisis previo o conocimiento especializado.
- IDE: *Software* de desarrollo.
- Intervalo de confianza: Rango de valores que contienen a una media.
- JMeter: Herramienta de testeo.
- Media: Promedio de los tiempos de respuesta generados por un factor o nivel.
- Media cuadrática: Valor que indica la variabilidad de un factor o nivel.
- Módulo Web: Parte de un sistema Web con funciones específicas.
- Significancia estadística: Una prueba de hipótesis indica que es muy poco probable que la misma haya ocurrido en virtud de las probabilidades.
- Spring *Framework*: Marco de trabajo modular.
- Struts *Framework*: Marco de trabajo encapsulado.
- Tasa de solicitud: Valor que mide la cantidad de peticiones que recibe la aplicación.
- Tasa de conversión: Valor que mide el logro de los objetivos para los cuales se desarrolló la aplicación. Ej: Del total de visitantes, cuántas realizaron una operación dentro de la aplicación.
- Tiempo de ejecución: Tiempo que toma un determinado programa dentro de un sistema operativo.
- Tiempo de respuesta: Tiempo en obtener datos o información desde el servidor.
- Univariante: Modelo con una variable dependiente y uno o más factores que actúan sobre ella.
- Valor crítico o significancia: Conjunto de valores que apoyan el rechazo de la hipótesis nula.

2.4 Marco referencial

En la región de Piura se pueden encontrar numerosas empresas de desarrollo de aplicaciones Web, en su mayoría estas empresas se abocan al desarrollo de *software* bajo pedido, no teniendo una rama en específico, según la encuesta realizada por Ramírez (2017), las empresas desarrollan en mayor grado los módulos Web de trámite documentario, ya sea para uso propio o para ser vendidos a otras empresas quienes llevan a cabo tal proceso, entre las empresas encuestadas se encuentran:

- Entercomp SAC
- Centro de Informática y Telecomunicaciones (CIT) de la Universidad Nacional de Piura.
- Registro Nacional de Identificación y Estado Civil (RENIEC) - Piura
- Caja Paita
- Caja Sullana
- Gobierno regional de Piura
- Municipalidad de Morropón – Chulucanas
- Municipalidad de Piura
- Municipalidad de Castilla

2.5 Hipótesis

2.5.1 Hipótesis general

Existe diferencia significativa en el tiempo de respuesta de los módulos Web desarrollados con los *frameworks* Spring y Struts 2.

2.5.2 Operacionalización de variables

Variables

- X1: Usuarios concurrentes
  I1X1: Número de usuarios concurrentes.

- X2: Tiempo de respuesta
  I2X2: Tiempo de respuesta promedio
  Para mayor detalle ver **Tabla 2.1**.

Tabla 2.1. Operacionalización de variables

Variables	Definición Conceptual	Definición Operacional	Dimensiones	Indicadores	Instrumentos
Usuarios concurrentes	Usuarios concurrentes son definidos como el número total de personas usando un recurso en línea en una ubicación particular dentro de un periodo de tiempo predefinido. (Lee, 2015)	Se determinaron en base a las características de los módulos Web desarrollados, *frameworks* y servidores Web.	Usuarios concurrentes base	Número de usuarios concurrentes	Guía de observación
Tiempo de respuesta	Tiempo de respuesta es el lapso de tiempo entre el fin de una petición en un sistema informático y el inicio de la recepción de la respuesta. (International Business Machines Corporation, 1993)	Tr = Latencia de red + Latencia Spring + Tiempo de acceso a la Base de datos Tr = Latencia de red + Latencia Struts 2 + Tiempo de acceso a la Base de datos	- Tiempos de respuesta del módulo Web desarrollado con Spring. - Tiempos de respuesta del módulo Web desarrollado con Struts 2.	- Tiempo de respuesta promedio	Guía de observación

III. MARCO METODOLÓGICO

3.1 Enfoque y diseño

La presente investigación fue desarrollada bajo un enfoque cuantitativo, siendo el diseño de investigación experimental; ya que se evaluó el rendimiento en cuanto a tiempo de respuesta del módulo Web bajo distintas circunstancias y *frameworks*, según Hernández, Fernández y Baptista (2014): "Los experimentos manipulan tratamientos, estímulos, influencias o intervenciones (denominadas variables independientes) para observar sus efectos sobre otras variables (las dependientes) en una situación de control".

Fue de nivel explicativa, cuya finalidad es determinar el efecto en el tiempo de respuesta (variable dependiente) debido a las variaciones del número de usuarios concurrentes, el tipo de operación y el nivel de los *frameworks* (variables independientes), como manifiesta Hernández, Fernández y Baptista (2014):

> "Van más allá de la descripción de conceptos o fenómenos o del establecimiento de relaciones entre conceptos; es decir, están dirigidos a responder por las causas de los eventos y fenómeno. Como su nombre lo indica, su interés se centra en explicar por qué ocurre un fenómeno y en qué condiciones se manifiesta o por qué se relacionan dos o más variables".

El tipo de investigación fue aplicada tecnológica; ya que generó conocimientos al sector de construcción de *software* y sirvió de guía para tomar una decisión de acuerdo al módulo Web considerado.

Para el desarrollo del análisis comparativo se empleó un diseño factorial con tres factores. Según Hernández, Fernández y Baptista (2014): "Los diseños factoriales manipulan dos o más variables independientes e incluyen dos o más niveles o modalidades de presencia en cada una de las variables independientes".

En el diseño factorial se han considerado factores que afectan el tiempo de respuesta del módulo Web, obteniéndose lo mostrado en la Tabla 3.1.

Tabla 3.1. Diseño factorial

Factor	Niveles
A: *Framework* de lado Servidor.	Spring *Framework* Struts 2 *Framework*
B: Tipo de operación	Autenticación Registro Consultas
C: Número de usuarios concurrentes	NUC 2*NUC 3*NUC

NUC: Número de usuarios concurrentes que hacen uso del módulo Web.

El diseño factorial de esta investigación será de 2x3x3.

3.2 Sujetos de la investigación

La presente investigación no contó con sujetos de investigación, en su lugar, se contó con elementos de investigación, siendo representados por los tiempos de respuesta obtenidos al efectuar las pruebas de rendimiento en los módulos Web desarrollados. Para obtener los tiempos de respuesta, se llevaron a cabo pruebas de rendimiento sobre los módulos Web desarrollados con los *frameworks* Spring y Struts, para ello se requirió contar con la interacción de una determinada cantidad de usuarios concurrentes. Los usuarios concurrentes utilizados en las pruebas de rendimiento y que permitieron el desarrollo del análisis comparativo fueron usuarios simulados virtualmente con la herramienta de testeo Jmeter, definiendo una cantidad base, la cual fue duplicada y triplicada, esto con el fin de realizar el planteamiento del diseño factorial empleado en la presente investigación, el cual consta de 3 niveles para el factor *Usuarios Concurrentes*.

3.3 Métodos y procedimientos

3.3.1 Parámetros de desarrollo del módulo Web

En base a la metodología RUP, se llevó a cabo el modelamiento de los parámetros inmersos en el proceso de trámite documentario representado de forma gráfica en la Imagen 3.1, necesarios para la elaboración de los módulos Web desarrollados con los *frameworks* Spring y Struts 2.

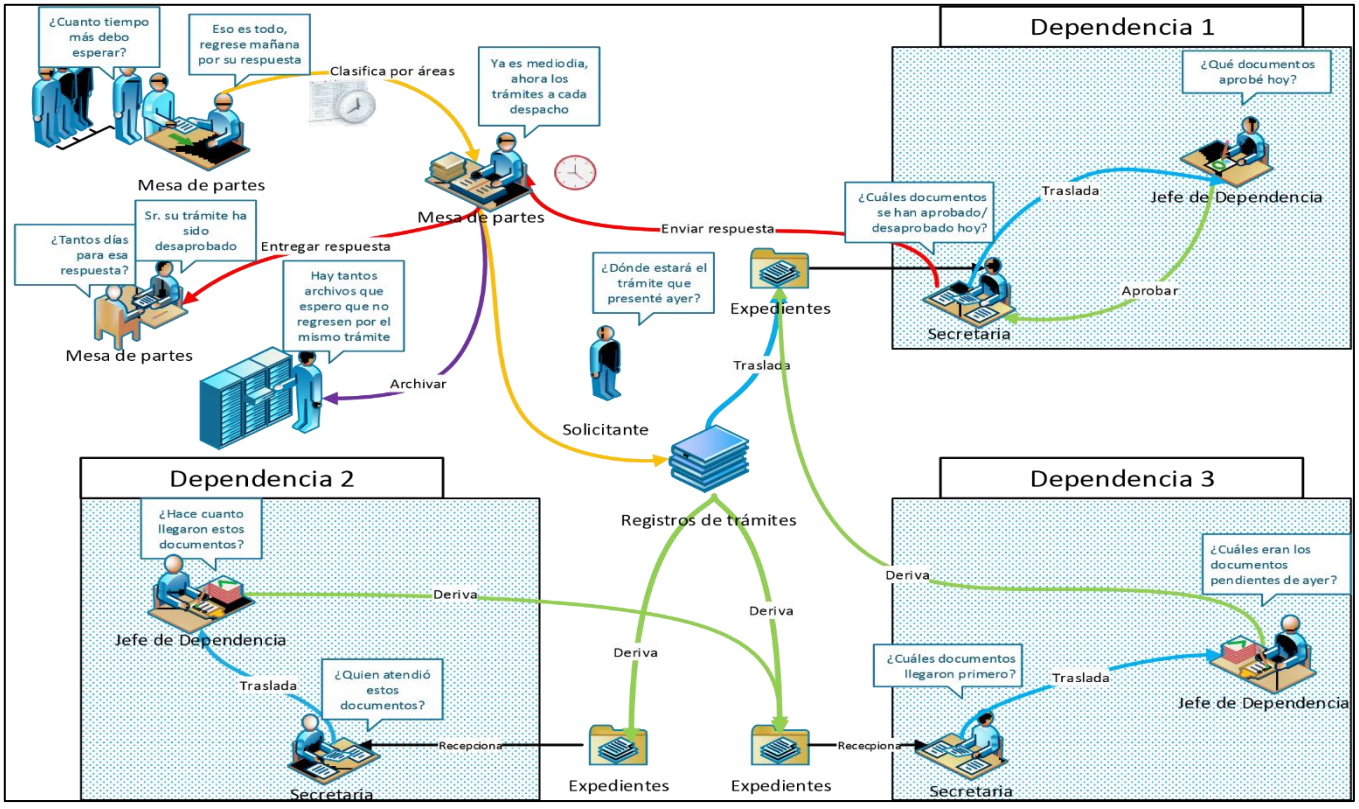


Figura 3.1. Proceso de trámite documentario en una organización.
Fuente: Proceso de trámite documentario (Calmet, 2014)

En la figura 3.1, se representan a las distintas unidades dentro de la institución como "dependencias", cada una de las dependencias atiende una determinada petición ya sea de un solicitante o de otra dependencia, cuando se realiza una solicitud entre dependencias, el expediente en cuestión es derivado completamente a una nueva unidad, la cual se encargará de completar la documentación que se

le indique, cuando no se requiera la realización de derivaciones a otras unidades, la unidad que tenga en el momento el expediente, lo derivará a mesa de partes, indicando que el expediente fue atendido y está listo para ser entregado al solicitante. Cabe resaltar que el expediente no retorna a una unidad previa, todas las unidades incluyen los documentos solicitados a su unidad y lo derivan, evitando que el expediente tenga un recorrido cíclico.

De este proceso, se determinaron los requerimientos funcionales, que son indicados en la Tabla 3.2 y cada uno de ellos es detallado en las tablas posteriores.

Tabla 3.2. Requerimientos funcionales

Item	Requerimientos
RF01	Permite a los usuarios consultar la bandeja de expedientes enviados.
RF02	Permite a los usuarios consultar la bandeja de expedientes recibidos.
RF03	Permite a los usuarios consultar la bandea de expedientes recepcionados.
RF04	Permite a los usuarios consultar la bandeja de expedientes finalizados.
RF05	Permite a los usuarios ver los movimientos de un expediente mediante el número de expediente.
RF06	Permite al solicitante consultar el movimiento de sus expedientes mediante el número de expediente.
RF07	Permite generar un reporte diario de los expedientes recepcionados por mesa de partes.
RF08	Permite a los usuarios registrar un trámite.

Fuente: Requerimientos funcionales (Ramírez, 2017)

Tabla 3.3. Especificación del requerimiento RF01

Consulta de bandeja de expedientes enviados	
Actor	Mesa de parte, Jefe de unidad.
Descripción	Permite a los usuarios consultas de los expedientes enviados.
Flujo básico	Los usuarios ingresan a la opción bandeja de enviados, se muestra una pantalla donde se listas los expedientes enviados.
Flujos alternos	No se han registrado trámites. Muestra en pantalla un mensaje indicando que no hay expedientes enviados.
Pre-condiciones	Los usuarios deben haber ingresado a la opción bandeja enviados.
Post-condiciones	Muestra una lista de los expedientes enviados por la oficina.

Fuente: Especificación de requerimientos (Ramírez, 2017)

Tabla 3.4. Especificación del requerimiento RF02

Consulta de bandeja de expedientes recibidos	
Actor	Mesa de parte, Jefe de unidad.
Descripción	Permite a los usuarios consultas de los expedientes recibidos.
Flujo básico	Los usuarios ingresan a la opción bandeja de recibidos, se muestra una pantalla donde se listan los expedientes recibidos.
Flujos alternos	No se han derivado trámites. Muestra en pantalla un mensaje indicando que no hay expedientes recibidos.
Pre-condiciones	Los usuarios deben haber ingresado a la opción bandeja recibidos.
Post-condiciones	Muestra una lista de los expedientes recibidos.

Fuente: Especificación de requerimientos (Ramírez, 2017)

Tabla 3.5. Especificación del requerimiento RF03

Consulta de bandeja de expedientes recepcionados	
Actor	Mesa de parte, Jefe de unidad.
Descripción	Permite a los usuarios consultas de los expedientes recepcionados.
Flujo básico	Los usuarios ingresan a la opción bandeja de recepcionados, se muestra una pantalla donde se listan los expedientes recepcionados.
Flujos alternos	No se han recepcionado expedientes. Muestra en pantalla un mensaje indicando que no hay expedientes recepcionados.
Pre-condiciones	Los usuarios deben haber ingresado a la opción bandeja recepcionados. El usuario debe haber recepcionado expedientes.
Post-condiciones	Muestra una lista de los expedientes recepcionados.

Fuente: Especificación de requerimientos (Ramírez, 2017)

Tabla 3.6. Especificación del requerimiento RF04

Consulta de bandeja de expedientes finalizados	
Actor	Mesa de parte, Jefe de unidad.
Descripción	Permite a los usuarios consultas de los expedientes finalizados.
Flujo básico	Los usuarios ingresan a la opción bandeja de finalizados, se muestra una pantalla donde se listan los expedientes finalizados.
Flujos alternos	No se han finalizado expedientes. Muestra en pantalla un mensaje indicando que no hay expedientes finalizados.
Pre-condiciones	Los usuarios deben haber ingresado a la opción bandeja finalizados. El usuario debe haber finalizado expedientes.
Post-condiciones	Muestra una lista de los expedientes finalizados.

Fuente: Especificación de requerimientos (Ramírez, 2017)

Tabla 3.7. Especificación del requerimiento RF05

Consultar movimientos de un expediente	
Actor	Mesa de parte, Jefe de unidad.
Descripción	Permite a los usuarios ver los movimientos de un determinado expediente.
Flujo básico	El solicitante ingresa a la opción ver movimientos. Se muestra una pantalla donde indica que ingrese el número de expediente, el usuario ingresa los datos y presiona el botón buscar.
Flujos alternos	No se encuentra registrado el número de expediente. Muestra en pantalla un mensaje indicando que el expediente buscado con el número de expediente ingresado no se encuentra.
Pre-condiciones	Los usuarios deben haber accedido a la opción ver movimientos.
Post-condiciones	Muestra los movimientos de expediente buscado.

Fuente: Especificación de requerimientos (Ramírez, 2017)

Tabla 3.8. Especificación del requerimiento RF06

Consultar movimientos de un expediente	
Actor	Solicitante
Descripción	Permite al solicitante o remitente ver los movimientos de un determinado expediente.
Flujo básico	El solicitante ingresa a la página de consultas de expedientes. Se muestra una pantalla donde indica que ingrese el número de expediente, el solicitante ingresa los datos y presiona el botón buscar.
Flujos alternos	No se encuentra registrado el número de expediente. Muestra en pantalla un mensaje indicando que el expediente buscado con el número de expediente ingresado no se encuentra.
Pre-condiciones	El solicitante debe haber accedido a la página de consultas de expedientes.
Post-condiciones	Muestra los movimientos del expediente buscado.

Fuente: Especificación de requerimientos (Ramírez, 2017)

Tabla 3.9. Especificación del requerimiento RF07

Reporte diario de los expedientes recepcionados	
Actor	Mesa de parte, Jefe de unidad
Descripción	Permite generar un reporte diario de los expedientes que se han registrado.
Flujo básico	El usuario ingresa a la opción de reporte diario, se muestra en pantallas un listado de los trámites registrados en el día.
Flujos alternos	No hay trámites registrados. Muestra en pantalla que no se han registrado trámites.
Pre-condiciones	El usuario debe haber accedido a la opción reporte diario.
Post-condiciones	Se genera satisfactoriamente un reporte.

Fuente: Especificación de requerimientos (Ramírez, 2017)

Tabla 3.10. Especificación del requerimiento RF08

Registrar trámite	
Actor	Mesa de parte, Jefe de unidad
Descripción	Permite registrar un trámite.
Flujo básico	Los usuarios acceden a la opción registrar trámite, se muestra una pantalla donde se debe indicar el tipo de trámite, remitente o solicitante, folio, asunto, unidad y prioridad. El usuario llena todos los datos y presiona el botón de registrar.
Flujos alternos	Solicitante o remitente no se encuentra registrado Muestra un mensaje indicando que no se encuentra el solicitante buscado.
Pre-condiciones	El usuario debe haber accedido a la opción registrar expediente.
Post-condiciones	Muestra mensaje de confirmación indicando que el trámite se ha registrado correctamente.

Fuente: Especificación de requerimientos (Ramírez, 2017)

Con el modelado del negocio y los requerimientos funcionales definidos, se muestra en la Figura 3.2, el proceso de trámite documentario.

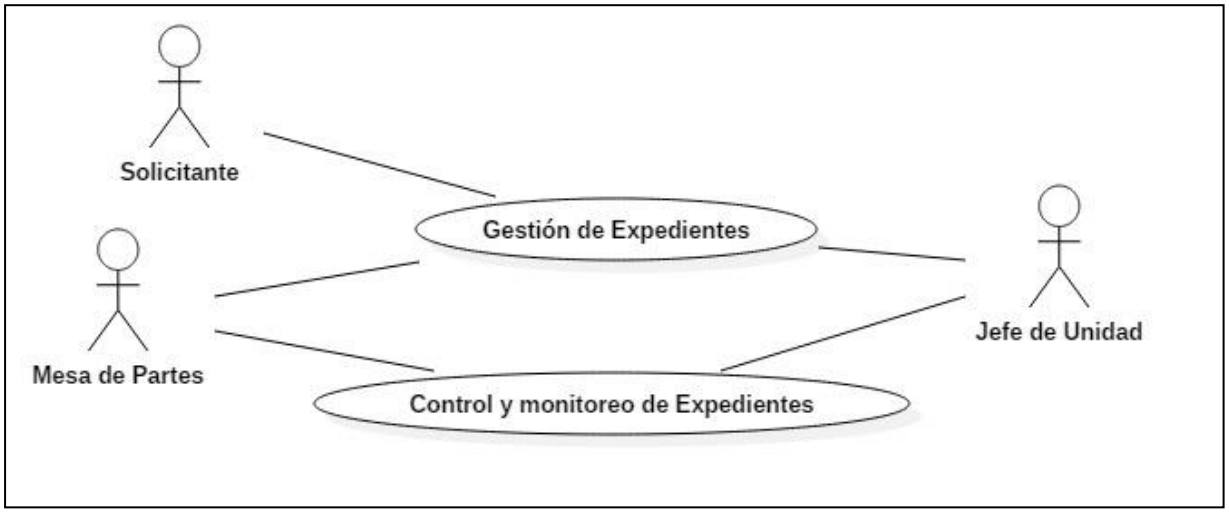


Figura 3.2. Proceso de trámite documentario.
Fuente: Diagrama de caso de uso del proceso de trámite documentario (Ramírez, 2017)

Se identificaron los actores que intervienen en el proceso, los cuales fueron detallados en la Tabla 3.11.

Tabla 3.11. Actores del proceso

Nombre	Descripción
Solicitante	Persona natural o jurídica, que se acerca a la entidad a realizar cualquier trámite. Del mismo modo, las unidades pueden comportarse como solicitante al emitir un pedido a otra unidad, a esto se le considera como trámite interno.
Mesa de partes	Usuario encargado de la recepción, registro y derivación de expedientes.
Jefe de unidad	Usuario encargado de gestionar los expedientes que recibe en su unidad.

Fuente: Especificación de requerimientos (Ramírez, 2017)

Con respecto a los diagramas de Casos de Uso, en base al modelado de negocio mostrado en la Figura 3.2, se dividieron en dos procesos o escenarios principales:

- Gestión de expedientes
- Control y monitoreo de expedientes.

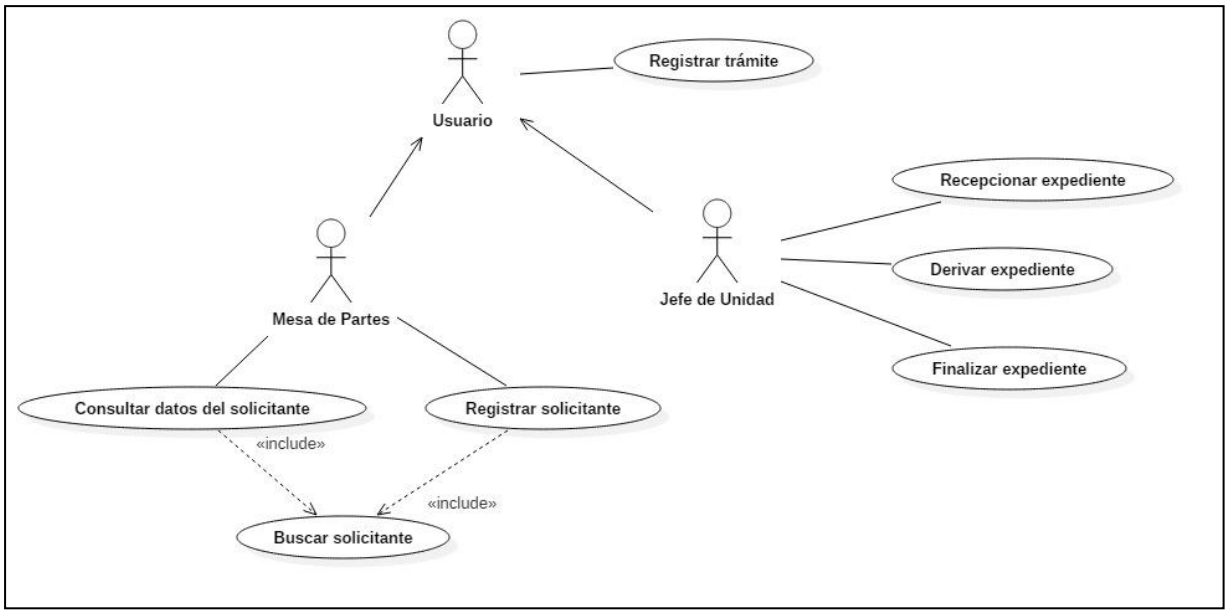


Figura 3.3. Gestión de expedientes.
Fuente: Diagrama de Caso de Uso de Gestión de Expedientes (Ramírez, 2017)

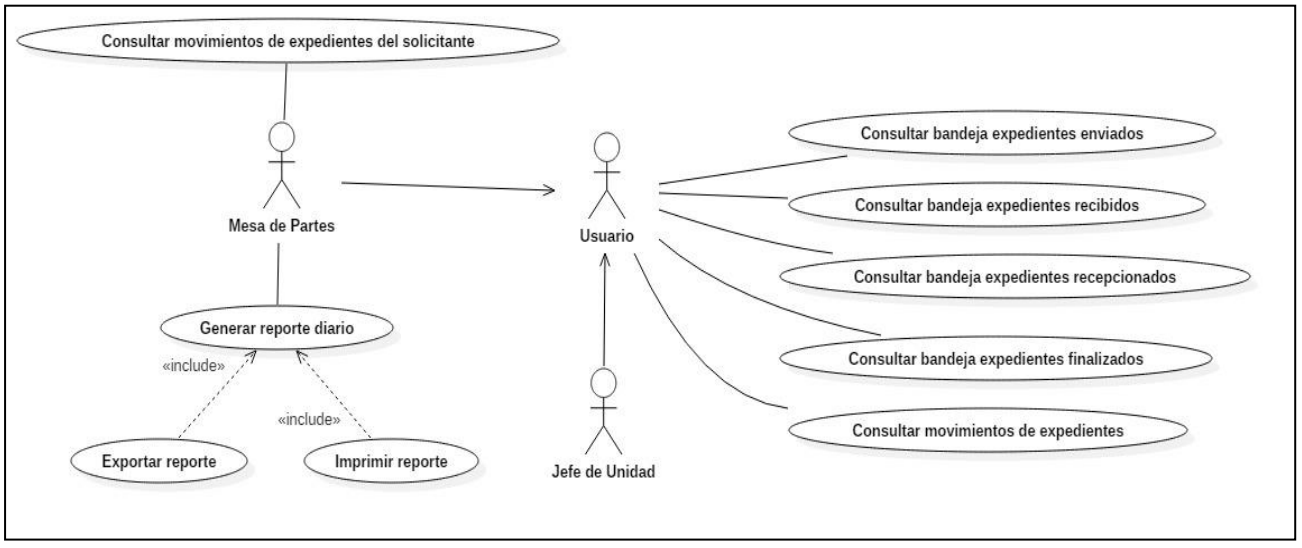


Figura 3.4. Control y monitoreo de expedientes.
Fuente: Diagrama Caso de Uso de Control y Monitoreo de Expedientes (Ramírez, 2017)

Los diagramas de secuencia que fueron obtenidos del modelado de la interacción de los objetos en el sistema, se muestran en las Figuras siguientes, indicando en letras rojas y líneas punteadas los menajes que se retornan al usuario si ocurre algún suceso no esperado o no se encuentra la información solicitada.

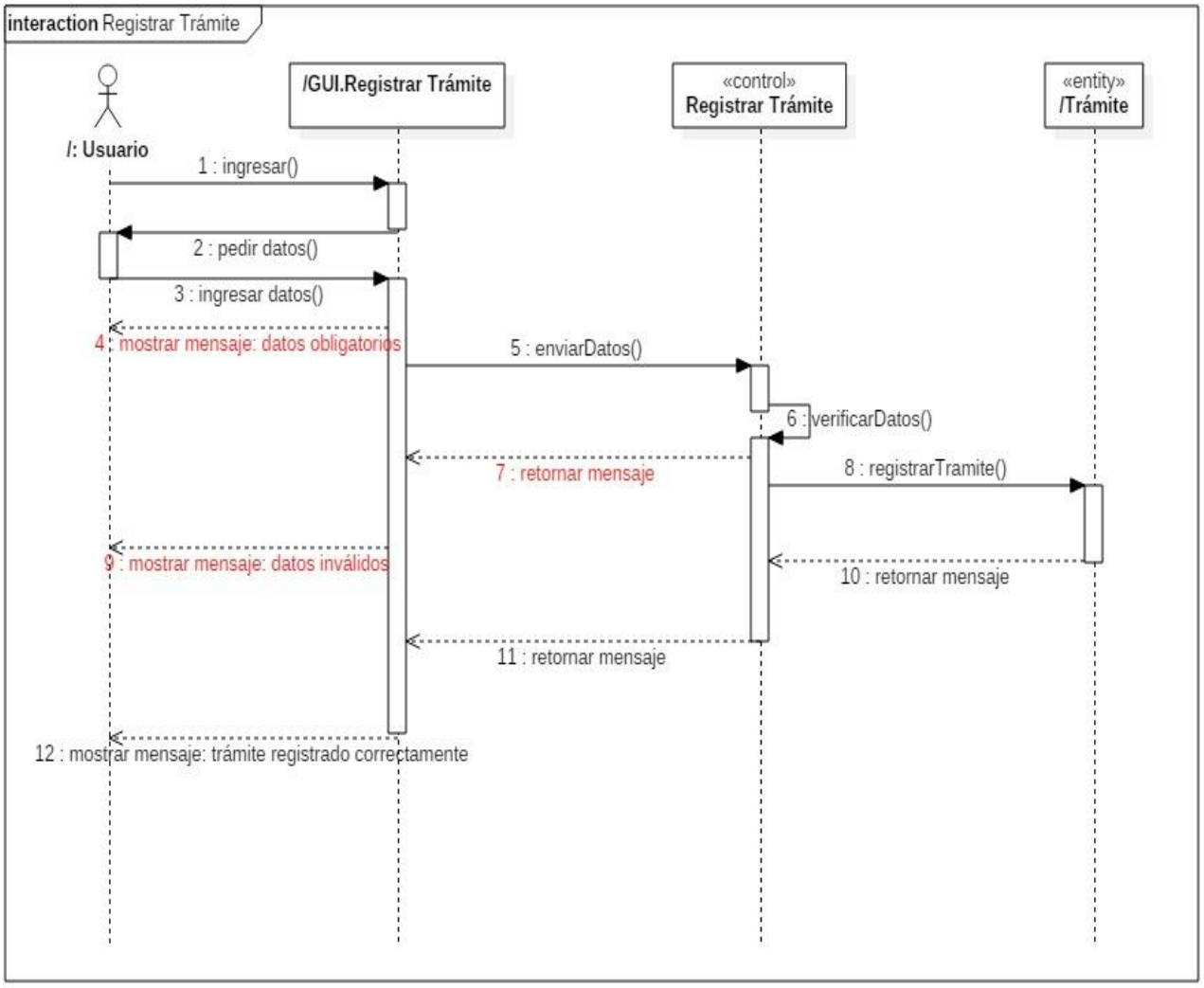


Figura 3.5. Registrar trámite.
Fuente: Diagrama de Secuencia – Registrar trámite (Ramírez, 2017)

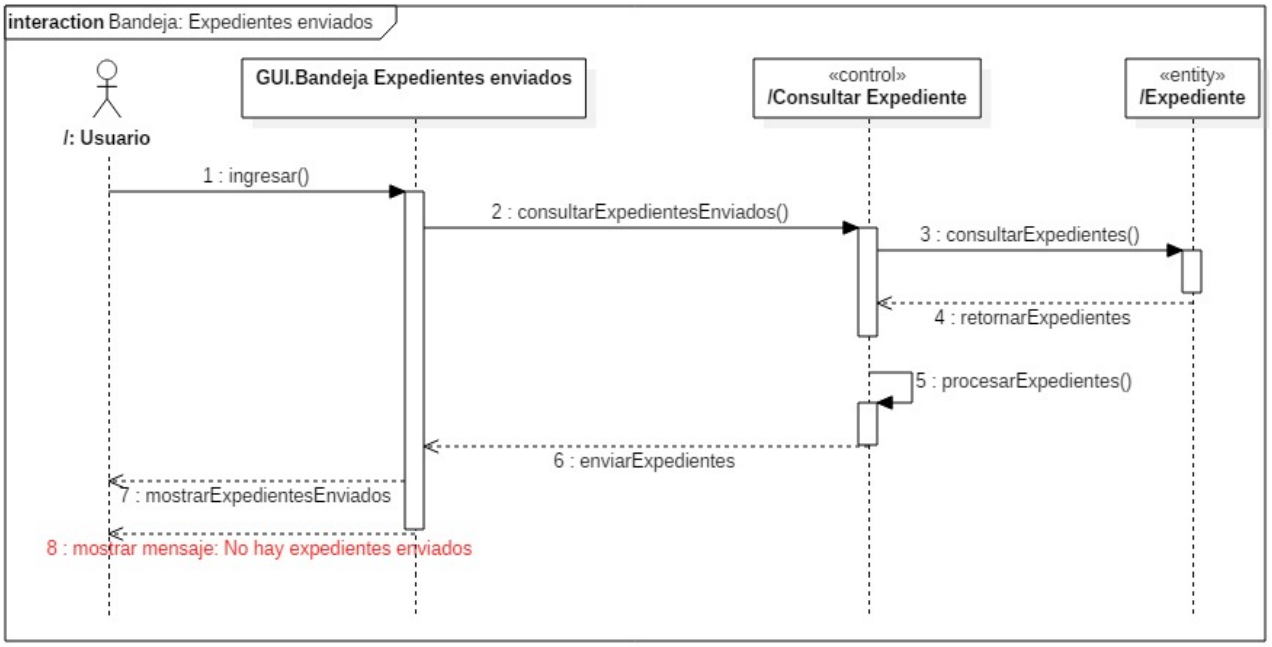


Figura 3.6. Consultar Bandeja de expedientes enviados.
Fuente: Diagrama de Secuencia – Bandeja de expedientes enviados (Ramírez, 2017)

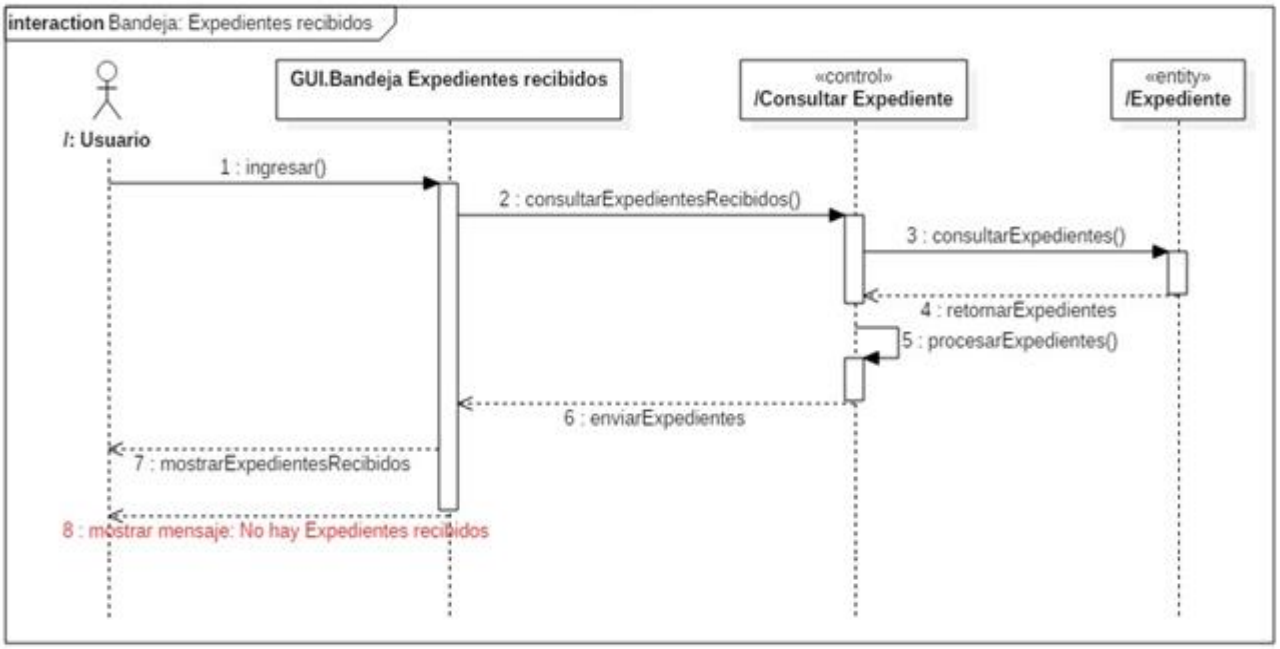


Figura 3.7. Consultar Bandeja de expedientes recibidos.
Fuente: Diagrama de Secuencia – Bandeja de expedientes recibidos (Ramírez, 2017)

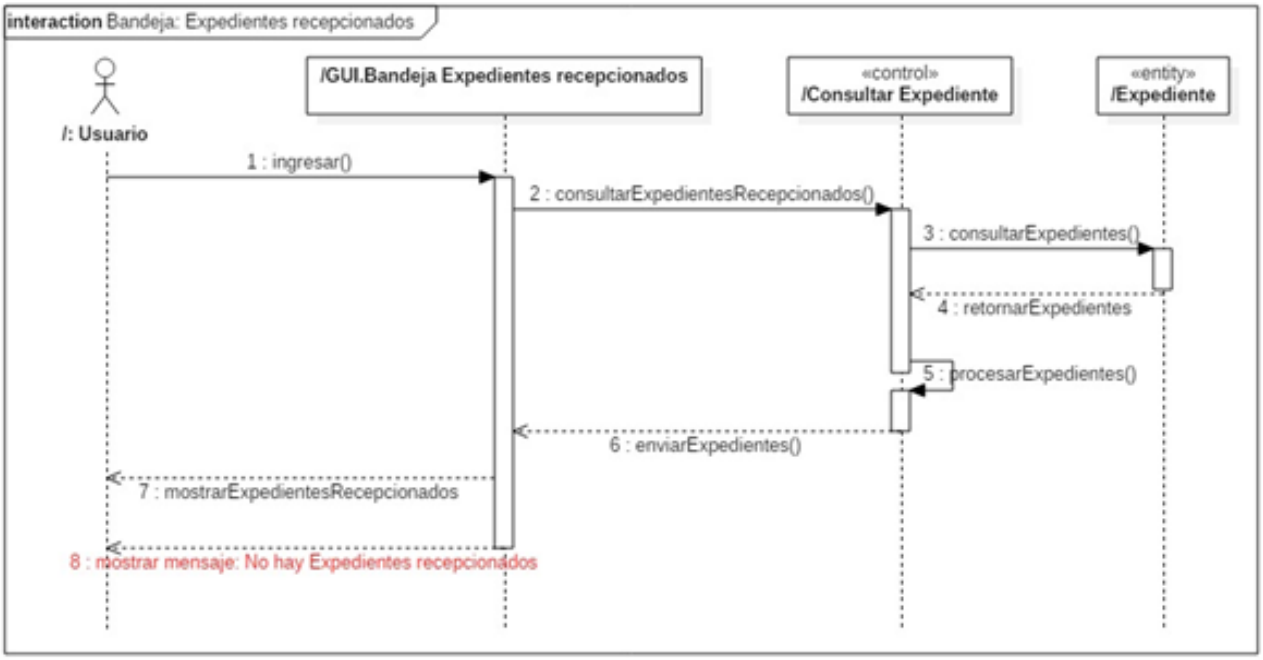


Figura 3.8. Consultar Bandeja de expedientes recepcionados.
Fuente: Diagrama de Secuencia – Bandeja de expedientes recepcionados (Ramírez, 2017)

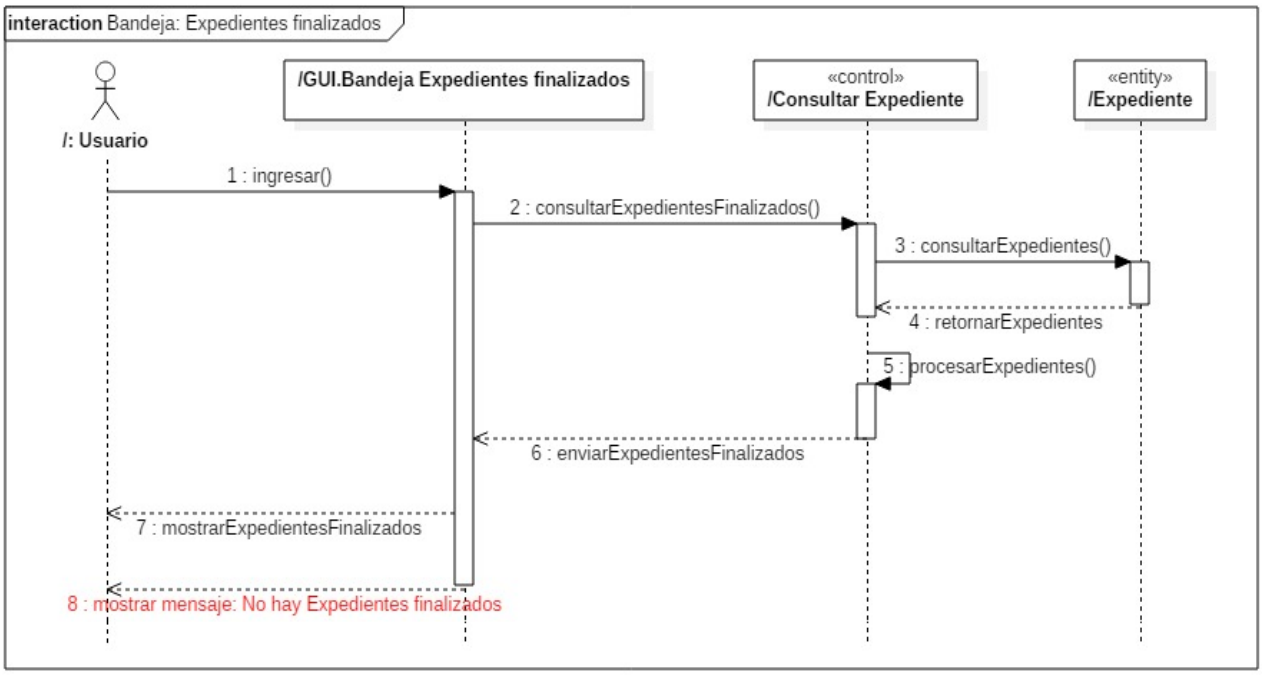


Figura 3.9. Consultar Bandeja de expedientes finalizados.
Fuente: Diagrama de Secuencia – Bandeja de expedientes finalizados (Ramírez, 2017)

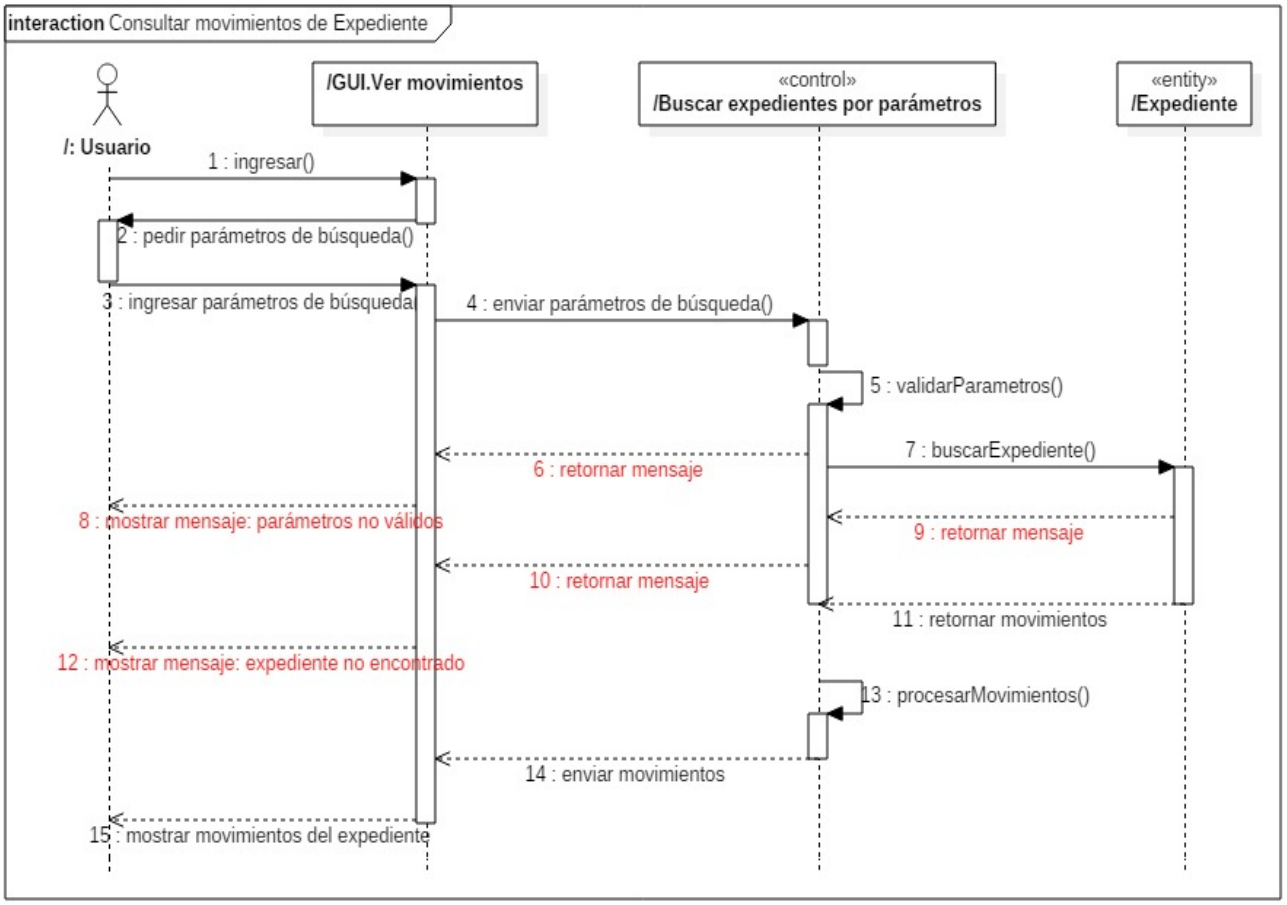


Figura 3.10. Consultar movimientos de expediente.
Fuente: Diagrama de Secuencia – Consultar movimientos de expedientes (Ramírez, 2017)

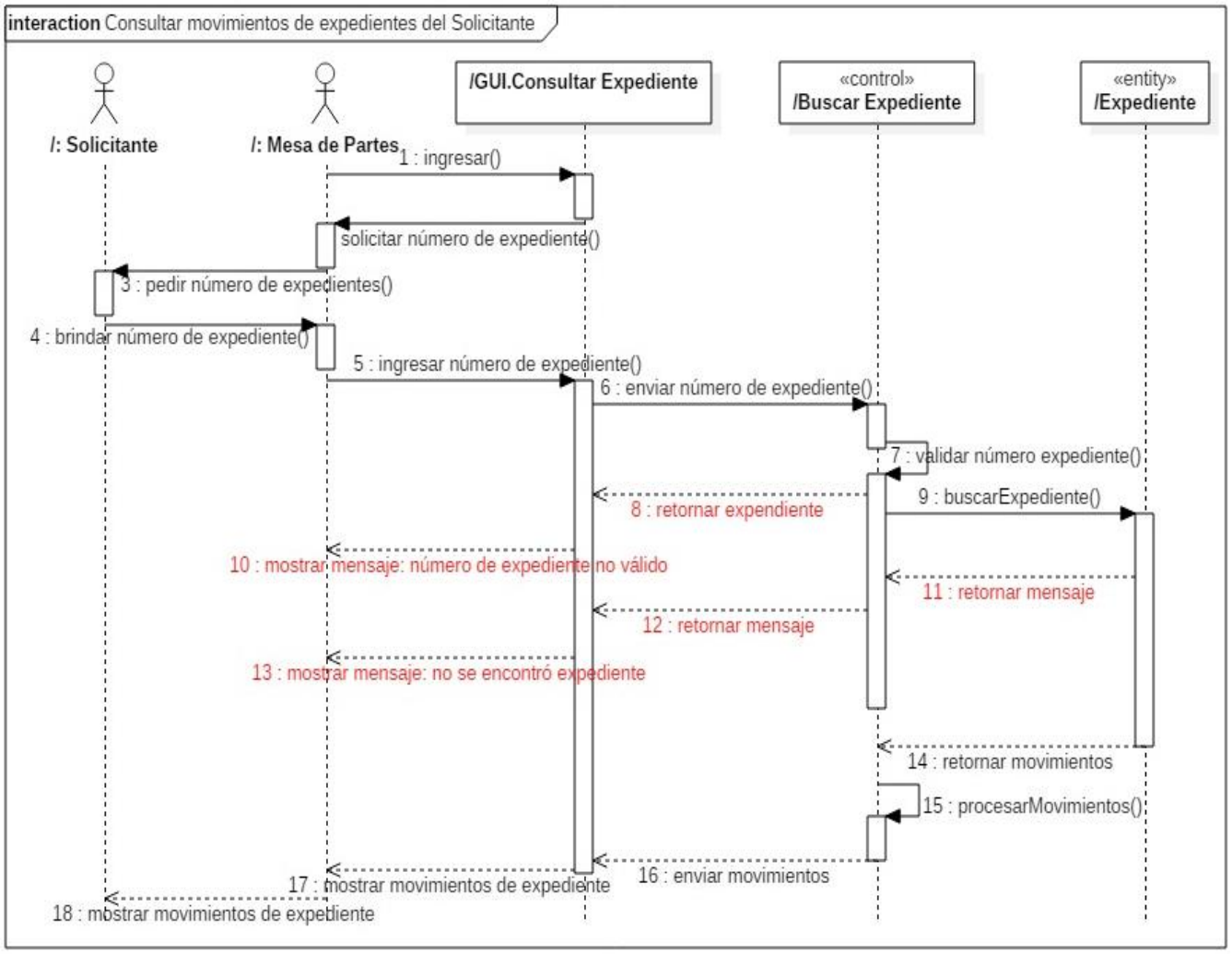


Figura 3.11. Consultar movimientos de expediente del Solicitante.
Fuente: Diagrama de Secuencia – Consultar movimientos de expediente del Solicitante (Ramírez, 2017)

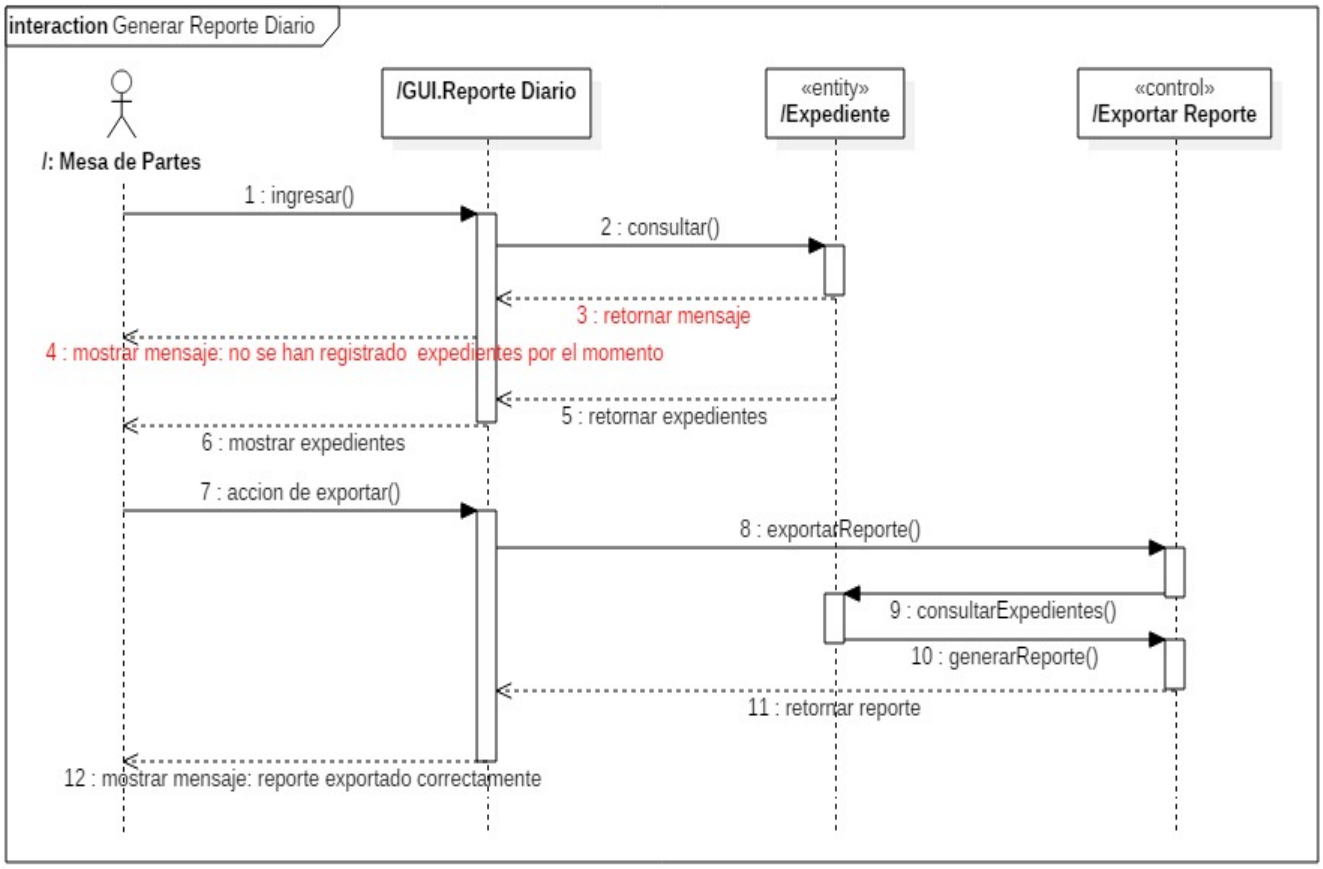


Figura 3.12. Generar reporte diario.
Fuente: Diagrama de Secuencia – Generar reporte diario (Ramírez, 2017)

El diagrama de clases que se obtuvo, producto del modelamiento de los requerimientos funcionales obtenidos de las diferentes instituciones consultadas, se muestra en la Figura 3.13.

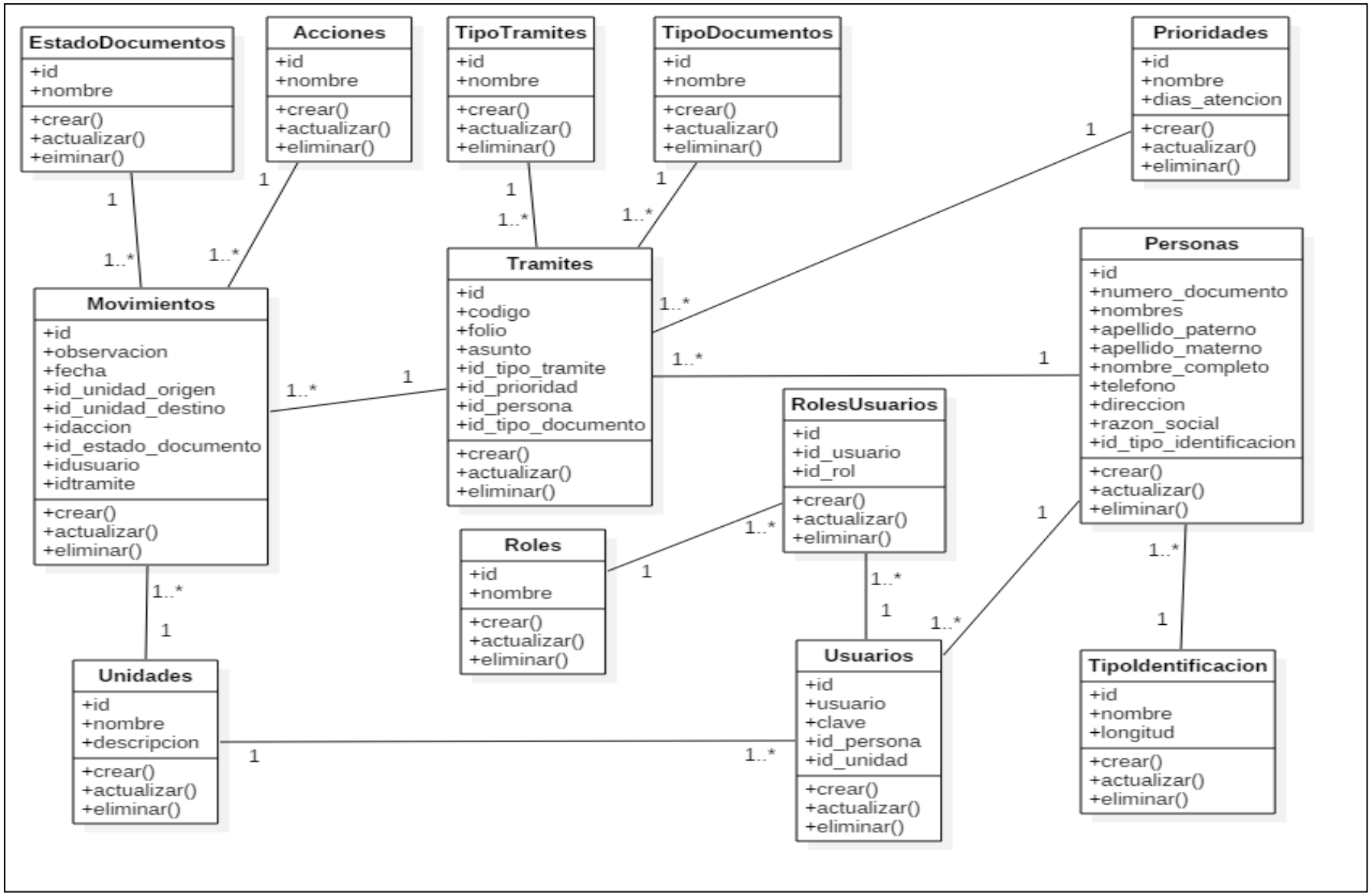


Figura 3.13. Diagrama de clases
Fuente: Diagrama de clases (Ramírez, 2017)

3.3.2 Definición y ejecución de pruebas

Las pruebas fueron realizadas con el *software* JMeter, el cual genera un reporte en formato HTML conteniendo los datos unitarios de cada petición realizada durante la prueba, así como el promedio de los tiempos.

Para llevar a cabo las pruebas, se debió fijar ciertos valores, necesarios para la correcta ejecución de las pruebas, el primer valor en ser fijado fue la cantidad de usuarios concurrentes base a ser usada, según *The Apache Software Foundation* (2017) indica "se debe elegir correctamente el número de usuarios a emplear, para evitar una sobrecarga, tanto en el servidor como en la herramienta de testeo". Por lo tanto, el parámetro de mayor importancia fue *la cantidad de usuarios concurrentes* a emplear, dado que se llevaron a cabo diferentes testeos, cada uno de ellos con un número de usuarios concurrentes distinto pero tomando en cuenta un número base.

Las consideraciones tomadas en cuenta fueron:

- Las cargas de trabajo que podría tener el módulo bajo una situación real (ONGEI, 2014).
- Las cargas de trabajo que podría tener el módulo bajo una situación real extrema (ONGEI, 2014).
- Simular un número de usuarios lo suficientemente alto como para observar las diferentes reacciones que podrían presentar los módulos.
- Los límites de carga que soporta el servidor de aplicaciones.
- Los límites de carga que soporta la máquina que aloja el servidor de aplicaciones.

Dadas las condiciones citadas, se observó que 500 usuarios concurrentes como base, teniendo en cuenta que en base al modelo factorial empleado en el desarrollo de la presente investigación, se presentan dos variaciones adicionales de los usuarios concurrentes, un factor multiplicativo de dos y tres veces la cantidad de usuarios concurrentes base, resultando en el uso de 500, 1000, y 1500 usuarios concurrentes empleados para las pruebas, y dado que serán manejados los resultados de forma estadística, se deben llevar a cabo tres repeticiones del experimento.

Con una base de 500 usuarios concurrentes, se debió fijar el tiempo que le tomará a los 500 usuarios concurrentes acceder al sitio, es decir, en cuantos segundos el total de usuarios deben haber ingresado al sistema, dado que se trabajó con un módulo de trámite documentario, se definió cinco usuarios por segundo, es decir, para 500 usuarios se debe tomar 100 segundos, para 1000 usuarios se debe tomar 200 segundos y para 1500 usuarios se debe tomar 300 segundo. El motivo por el que se definieron cinco usuarios por segundo se debe a la simulación realizada, se llevó a cabo con cinco unidades, cada unidad contaba con cinco ventanillas de atención a los clientes, atendiendo a un usuario por segundo, a su vez, es un valor que permite obtener datos sin llegar a saturar el servidor de aplicaciones.

Las especificaciones del *software* que fue utilizado en la realización de las pruebas se indican en la Tabla 3.12 y las especificaciones de las computadoras que fueron empleadas, todas del mismo modelo, se encuentran en la Tabla 3.13.

Tabla 3.12. Especificaciones de software

Tipo	Software
Sistema operativo	Windows 10
Motor de base de datos	PostgreSQL 10.4 – 64 bits
Servidor de aplicaciones	Glassfish 5
Herramienta de testeo	Apache JMeter 5.0
Máquina virtual de Java	JDK 1.8u172
Entorno Java EE	Java EE 8

Tabla 3.13. Especificaciones de hardware

Componente	Descripción
Procesador	Intel Core I5 x64 2.1 GHz
Disco duro	1 TB
Memoria RAM	4 GB

Para llegar a determinar la base de 500 usuarios concurrentes, y asegurar un rendimiento óptimo por parte del servidor de aplicaciones al ejecutar las pruebas, se realizaron pruebas de estrés con valores arbitrarios usando el tercer factor multiplicativo, esto con el fin de poder determinar la cantidad máxima de usuarios que se podrían usar sin llegar a obtener respuestas erróneas por parte del servidor, se fijaron cantidades que, dependiendo de las especificaciones tanto de *hardware* y *software*, se pensó podrían ser soportadas, se realizaron pruebas con 1500, 2100 y 3000 usuarios concurrentes, como se muestra en las Figuras 3.15, 3.16, las pruebas realizadas empezaron a generar errores en las peticiones debido a la saturación del servidor de aplicaciones, aun siendo configurado para aceptar grandes cargas de trabajo, debido a las limitaciones mencionadas anteriormente, en la Figura 3.14, se puede observar que se completaron las peticiones, lo cual indicó que 1500 usuarios concurrentes fue la carga idónea para la realización de las pruebas posteriores.

```
[jmeter] Starting the test @ Sun Jul 29 00:40:05 COT 2018 (1532842805470)
[jmeter] Waiting for possible Shutdown/StopTestNow/Heapdump message on port 4445
[jmeter] summary +   8160 in 00:00:24 =  336,1/s Avg:    92 Min:     0 Max:  2998 Err:     0 (0,00%) Active: 62 Started: 106 Finished: 44
[jmeter] summary +  18754 in 00:00:30 =  626,7/s Avg:    47 Min:     0 Max:  3559 Err:     0 (0,00%) Active: 15 Started: 236 Finished: 221
[jmeter] summary =  26914 in 00:00:54 =  496,5/s Avg:    61 Min:     0 Max:  3559 Err:     0 (0,00%)
[jmeter] summary +  14731 in 00:00:30 =  490,1/s Avg:     3 Min:     0 Max:   446 Err:     0 (0,00%) Active: 1 Started: 360 Finished: 359
[jmeter] summary =  41645 in 00:01:24 =  494,2/s Avg:    40 Min:     0 Max:  3559 Err:     0 (0,00%)
[jmeter] summary +  13804 in 00:00:30 =  454,7/s Avg:     2 Min:     0 Max:   310 Err:     0 (0,00%) Active: 1 Started: 479 Finished: 478
[jmeter] summary =  55449 in 00:01:55 =  483,8/s Avg:    31 Min:     0 Max:  3559 Err:     0 (0,00%)
[jmeter] summary +  14268 in 00:00:30 =  477,8/s Avg:     3 Min:     0 Max:   590 Err:     0 (0,00%) Active: 1 Started: 602 Finished: 601
[jmeter] summary =  69717 in 00:02:24 =  482,5/s Avg:    25 Min:     0 Max:  3559 Err:     0 (0,00%)
[jmeter] summary +  14964 in 00:00:30 =  503,2/s Avg:     3 Min:     0 Max:   203 Err:     0 (0,00%) Active: 1 Started: 731 Finished: 730
[jmeter] summary =  84681 in 00:02:54 =  486,1/s Avg:    21 Min:     0 Max:  3559 Err:     0 (0,00%)
[jmeter] summary +  13997 in 00:00:30 =  466,8/s Avg:     3 Min:     0 Max:   190 Err:     0 (0,00%) Active: 3 Started: 851 Finished: 848
[jmeter] summary =  98678 in 00:03:24 =  483,2/s Avg:    19 Min:     0 Max:  3559 Err:     0 (0,00%)
[jmeter] summary +  14780 in 00:00:30 =  495,0/s Avg:     8 Min:     0 Max:   565 Err:     0 (0,00%) Active: 11 Started: 984 Finished: 973
[jmeter] summary = 113458 in 00:03:54 =  484,7/s Avg:    17 Min:     0 Max:  3559 Err:     0 (0,00%)
[jmeter] summary +  11123 in 00:00:30 =  370,9/s Avg:    96 Min:     1 Max:  1952 Err:     0 (0,00%) Active: 75 Started: 1113 Finished: 1038
[jmeter] summary = 124581 in 00:04:24 =  471,8/s Avg:    24 Min:     0 Max:  3559 Err:     0 (0,00%)
[jmeter] summary +  17777 in 00:00:30 =  592,5/s Avg:    76 Min:     1 Max:  1099 Err:     0 (0,00%) Active: 9 Started: 1232 Finished: 1223
[jmeter] summary = 142358 in 00:04:54 =  484,1/s Avg:    31 Min:     0 Max:  3559 Err:     0 (0,00%)
[jmeter] summary +  15812 in 00:00:30 =  526,8/s Avg:    11 Min:     0 Max:   461 Err:     0 (0,00%) Active: 20 Started: 1375 Finished: 1355
[jmeter] summary = 158170 in 00:05:24 =  488,1/s Avg:    29 Min:     0 Max:  3559 Err:     0 (0,00%)
[jmeter] summary +   4867 in 00:00:30 =  162,1/s Avg:   437 Min:     4 Max:  1999 Err:     0 (0,00%) Active: 126 Started: 1497 Finished: 1371
[jmeter] summary = 163037 in 00:05:54 =  460,4/s Avg:    41 Min:     0 Max:  3559 Err:     0 (0,00%)
[jmeter] summary +   3861 in 00:00:30 =  128,9/s Avg:  1005 Min:     3 Max:  4141 Err:     0 (0,00%) Active: 120 Started: 1500 Finished: 1380
[jmeter] summary = 166898 in 00:06:24 =  434,6/s Avg:    63 Min:     0 Max:  4141 Err:     0 (0,00%)
[jmeter] summary +   7102 in 00:00:14 =  498,5/s Avg:   191 Min:     0 Max:  1583 Err:     0 (0,00%) Active: 0 Started: 1500 Finished: 1500
[jmeter] summary = 174000 in 00:06:38 =  436,9/s Avg:    68 Min:     0 Max:  4141 Err:     0 (0,00%)
[jmeter] Tidying up ...    @ Sun Jul 29 00:46:44 COT 2018 (1532843204253)
[jmeter] ... end of run
```

Figura 3.14. Pruebas de estrés con 1500 usuarios concurrentes.

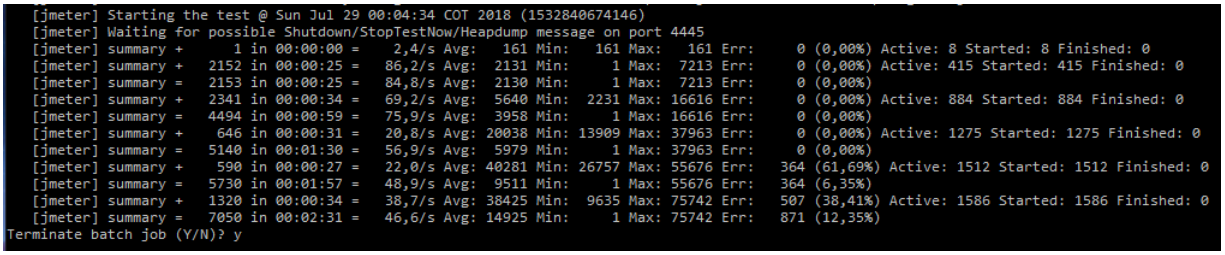


Figura 3.15. Pruebas de estrés con 2100 usuarios concurrentes.

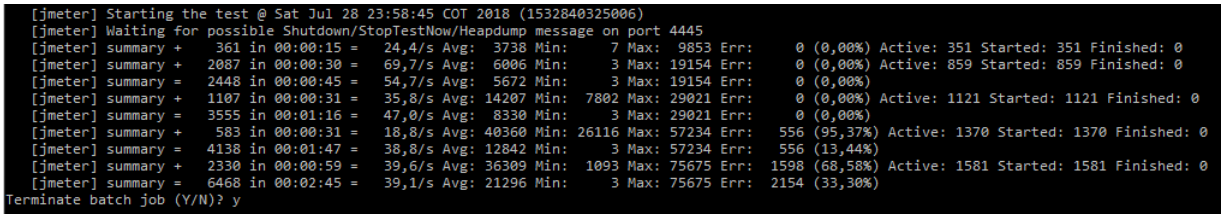


Figura 3.16. Pruebas de estrés con 3000 usuarios concurrentes.

Con los usuarios concurrentes y tiempo de subida definidos, se procedió a configurar la herramienta JMeter, a fin de poder generar el escenario que fue usado en las pruebas, para ello, JMeter cuenta con un elemento de configuración llamado *HTTP(S) Test Script Recorder*, el cual permite grabar la interacción que tiene un determinado usuario en una aplicación, esto lo logra cumpliendo la función de *proxy* y capturando todo el flujo de información que se genere, es por ello que debe ser configurado previamente si se requiere un filtrado de los elementos a capturar, la configuración empleada al momento de realizar la grabación del escenario se presenta en **ANEXO N° 15**.

La configuración del *HTTP(S) Test Script Recorder* consistió en definir un filtro básico para la petición de archivos con extensión 'txt', el resto de peticiones serían aceptadas y grabadas, también se indicó que se almacenaran las cabeceras HTTP y que se guardaran las redirecciones hechas con el fin de no alterar el flujo que sería grabado, un caso de ejemplo serían las redirecciones realizadas como resultado de la autenticación de los usuarios en el módulo Web, cada usuario tiene una función asignada dentro del módulo, por lo tanto, al ser autenticado, ciertos datos del usuario son agregados a una sesión dentro del servidor y es redireccionado a una página donde se le muestren la opciones que tiene a su disposición, por lo tanto el proceso de redirección es el resultado de una autenticación exitosa, lo cual es de interés para la realización de las pruebas, pues cada uno de los *frameworks* empleados en el desarrollo de cada módulo, gestionan a su manera, el acceso de a los datos almacenados en las sesiones del servidor, otro argumento referente a realizar el seguimiento de las redirecciones es que los datos puestos en la sesión del servidor serán usados en operaciones posteriores, por lo tanto, es vital que el proceso de autenticación se realice completamente.

La opción "*Target Controller*", la cual permite usar un elemento de grabación configurado dentro del grupo de hilos creado para el plan de pruebas, esta opción permite especificar cuál de todos los controladores será usado como destino y almacenar todas las peticiones obtenidas al momento de grabar el escenario, esta opción permite almacenar las peticiones según se requiera, por ejemplo, podrían separarse las peticiones hechas a un determinado módulo de una aplicación Web, como el módulo de gestión de Recursos Humanos y las peticiones hechas al panel de administración.

Se definió un "*Thread Group*", grupo de hilos, que contiene todos los elementos de configuración empleados en la grabación y ejecución de las pruebas a excepción del *HTTP(S) Test Script Recorder* descrito en párrafos anteriores, el cual es agregado al mismo nivel que el *Thread Group.*

El elemento *Thread Group* posee opciones de configuración que permiten controlar la ejecución del plan de pruebas, como se muestra en el **ANEXO N° 16**. Para las pruebas, el número de hilos (usuarios) y el periodo de subida (en segundos) fue variable, según lo explicado en párrafos anteriores; el contador de ciclos fue uno, es decir, el plan de prueba solo se ejecutaría una sola vez, para cada variante, no se especificó configuración alguna en el apartado de planificación, el plan seguiría el flujo fijado por el escenario que fue grabado con anterioridad. Se empleó la opción "continuar en error", lo indica a JMeter que el test no se detenga si existe algún error en las peticiones que se realicen al servidor de aplicaciones, con el fin de detectar errores en la ejecución de las operaciones programadas en los módulos Web.

Dentro del elemento *Thread Group*, se agregan los elementos de control que actúan mientras se ejecutan las pruebas, los elementos básicos de configuración empleados son mostrados en el **ANEXO N° 17**. El elemento *Recording Controller* permite almacenar las peticiones que son grabadas por el *HTTP*(S*) Test Script Recorder*, cada uno de los controladores que se agreguen poseerán distintas peticiones, lo cual permite ejecutar diferentes escenarios dentro de un mismo plan. Los elementos de configuración *HTTP Cookie Manager* y *HTTP Cache Manager* son elementos que simulan el accionar de un navegador Web, lo que permite trabajar con sesiones y la cache empleados en un escenario real, con la adición de los mencionados elementos se evitan errores al ejecutar las peticiones que se encuentran dentro de los controladores. El elemento *HTTP Request Defaults* permite sobrescribir los diferentes elementos empleados al realizar las peticiones HTTP, como el dominio al cual acceder, el protocolo a emplear, el número de puerto, la codificación a emplearse al enviar la petición, servidor *proxy*, parámetros a enviar, tiempo de espera, el manejo de las descargas, si se usará un nombre de dominio o un número IP e indicar el tipo de dirección IP a emplear. No se configuró elementos adicionales, como los elementos que recogen los datos de las muestras.

Con los elementos configurados, se procedió a la ejecución de las pruebas, pero no se hizo uso de la interfaz gráfica de JMeter, el motivo fue por temas de rendimiento de la propia herramienta, como se indica en el **ANEXO N° 06**, lo recomendado es emplear el modo consola al ejecutar planes de prueba que tengan una cantidad alta de datos a almacenar como resultado de la ejecución de la prueba.

La ejecución de las pruebas fue realizada el día martes 10 y 24 de Julio de 2018, según el diseño factorial, se tuvieron dos variantes de los módulos Web, un módulo con el *framework* Spring y otro con el *framework* Struts, tres variantes de los usuarios concurrentes, 500, 1000 y 1500 usuarios, y tres tipos de operaciones, se realizaron seis repeticiones de los planes de pruebas los cuales contenían las variantes de los *frameworks* y los usuarios concurrentes, realizándose un total de 36 ejecuciones, siguiéndose el esquema:

- Realizar la configuración y verificación del servidor de aplicaciones, servidor de base de datos, la herramienta JMeter, así como los ordenadores sobre los que se ejecutan, logrando asegurar un óptimo funcionamiento durante las pruebas.
- Realizar la configuración de los planes de pruebas para las tres variantes de usuarios concurrentes.
- Ejecutar un plan de prueba y anotar los datos necesarios en las guías de observación, las cuales pueden ser observadas en los **ANEXOS N° 07 al 12**.
- Verificar los resultados del plan de prueba ejecutado en búsqueda de errores en las peticiones, si existía algún error, se descartaba la prueba.
- Verificar el estado del servidor de aplicaciones y el servidor de base de datos, así como los ordenadores y el ordenador desde el cual eran ejecutadas las pruebas, si alguno presentaba deficiencia en su funcionamiento, se reiniciarían los servicios o el equipo si era necesario, con

el fin de obtener un rendimiento óptimo para evitar que factores externos afectaran los resultados de las pruebas.

- Ejecutar el siguiente plan de prueba hasta completar las 36 ejecuciones.

El proceso de la ejecución de las pruebas se muestra en la Figura 3.17.

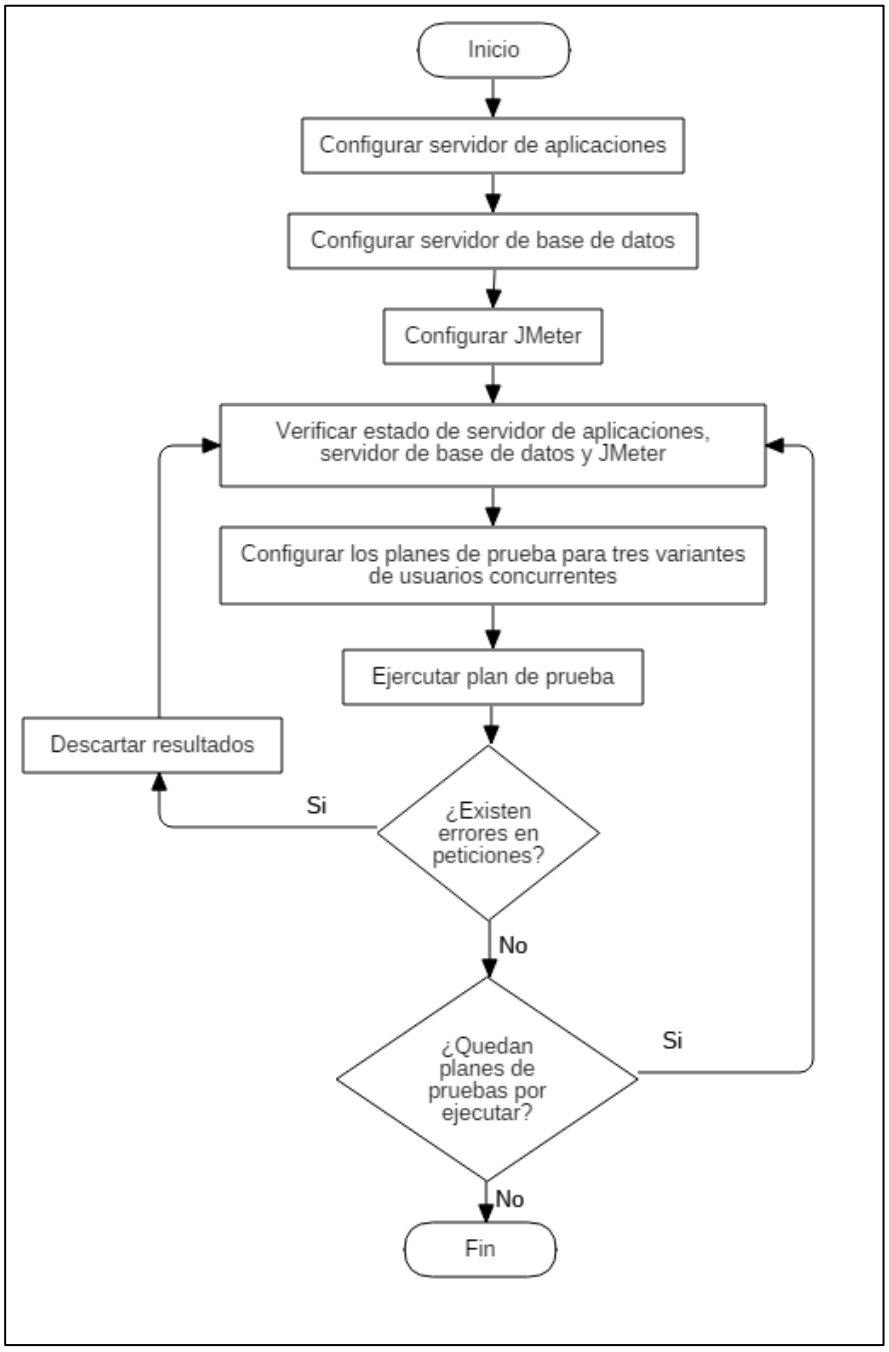


Figura 3.17. Diagrama de flujo del proceso de ejecución de pruebas.

Los resultados de la ejecución de las pruebas fueron exportados de forma automática a archivos HTML como reportes por la herramienta JMeter, en total fueron generados automáticamente 36 reportes, uno de ellos se muestra en el **ANEXO N° 13**, los resultados de estos reportes fueron filtrados para obtener las operaciones de interés para la presente investigación:

- Operación de registro.
- Operación de consulta
- Operación de autenticación

3.4 Técnicas e instrumentos

Técnica de muestreo: Muestreo por conveniencia.

Técnicas de recolección de datos: De laboratorio

Instrumentos de recolección de datos

- Guía de observación
- Apache JMeter

De análisis

- Análisis de varianza factorial – ANOVA
- Diferencia de medias estadísticas

Las validaciones de la guía de observación hecha por especialistas y que fue empleada en la presente investigación se muestran en el **ANEXO N° 5.**

La matriz general de consistencia se muestra en el **ANEXO N° 14.**

3.5 Aspectos éticos

Tanto los datos como los resultados obtenidos no fueron modificados durante el desarrollo de la presente investigación, así como el desarrollo del estudio plasmado es de total autoría del tesista. El estudio llevado a cabo fue realizado de manera neutral y objetiva, no se trató de favorecer o perjudicar a ninguno de los elementos que fueron objeto de estudio.

IV. RESULTADOS Y DISCUSIÓN

4.1 Resultados

Con los resultados obtenidos de la ejecución de los planes de prueba se construyó la Tabla 4.1, en la cual se plantearon los factores involucrados en el modelo factorial empleado en la presente investigación, el cual fue un modelo factorial de 2x3x3, el factor *Framework de lado servidor* con dos niveles (Spring y Struts), el factor *Tipo de Operación* con tres niveles (Autenticación, Registro y Consultas) y el factor *Usuarios concurrentes* con tres niveles (500, 1000 y 1500). En el factor *Framework de lado servidor* se omitieron las versiones de ambos *frameworks*.

En el desarrollo del presente capítulo, se emplearon nombres cortos para reemplazar a los nombres de los factores, para el factor *Framework de lado servidor* se empleó *Framework*, para el factor *Tipo de operación* se empleó *Operación*, para el factor *Número de usuarios concurrentes* se empleó *NroUsuariosConcurrentes* y los tiempos de respuesta fueron expresados en milisegundos (ms).

Tabla 4.1. Diseño factorial 2x3x3

***Framework* de lado servidor**	**Tipo de Operación**								
	Autenticación			**Registro**			**Consultas**		
	Número de Usuarios Concurrentes			**Número de Usuarios Concurrentes**			**Número de Usuarios Concurrentes**		
	500	**1000**	**1500**	**500**	**1000**	**1500**	**500**	**1000**	**1500**
Spring	5.40	9.90	11.50	5.40	6.70	9.70	3.61	8.02	11.30
	6.40	160.50	120.40	4.70	84.70	61.80	3.84	92.91	66.09
	169.90	263.40	1694.40	92.00	138.60	856.40	89.36	150.09	874.45
	189.50	265.10	1529.00	113.70	153.10	880.40	165.80	176.95	868.05
	190.50	253.60	1620.90	123.90	159.50	879.80	127.82	185.45	1178.64
	190.10	224.90	1164.10	120.30	183.20	922.30	187.14	268.16	1005.02
Struts	5.00	4.90	38.90	5.60	5.30	39.50	4.34	5.66	44.98
	135.10	349.50	195.10	131.30	345.80	197.50	135.66	363.27	207.64
	237.40	57.40	3105.57	237.80	61.20	3061.90	245.57	83.61	3041.00
	178.80	261.50	1669.30	149.70	260.90	1766.90	179.45	374.55	1572.43
	291.90	627.20	1534.70	179.70	585.20	1610.00	221.02	917.64	1588.32
	156.80	307.60	1350.60	176.50	389.00	1124.40	152.34	396.41	924.00

En la tabla 4.2 se muestra el total de datos que fueron obtenidos de las pruebas y fueron usados como los niveles de cada factor en las pruebas estadísticas posteriores.

Tabla 4.2. Factores y datos

Factores		N
Framework	Spring	54
	Struts	54
Operación	Autenticación	36
	Consulta	36
	Registro	36
NroUsuariosConcurrentes	500	36
	1000	36
	1500	36

Para el análisis estadístico de los datos se emplearon dos pruebas:

- Análisis de Varianza Factorial Univariante – ANOVA
- Diferencia de medias estadísticas

ANOVA permite evaluar el efecto de uno o más factores sobre la variable dependiente tanto de forma individual como conjunta, la prueba de diferencia de medias estadísticas permite evaluar si existe una diferencia significativa en las medias de dos muestras comparadas, ambas pruebas empleando un nivel de significancia del 0.05, es decir, un intervalo de confianza del 95%, esperando obtener como error máximo posible un 5%; en la presente investigación se planteó un tratamiento lineal de los datos, como se puede observar en la Figura 4.1, en la cual se aprecian las medias del conjunto de factores (grupos) del modelo factorial comparadas con la desviación estándar de los tiempos de respuesta (variable dependiente) y su comportamiento, el cual gráficamente tiene un comportamiento lineal, es decir, es posible aproximar a una recta.

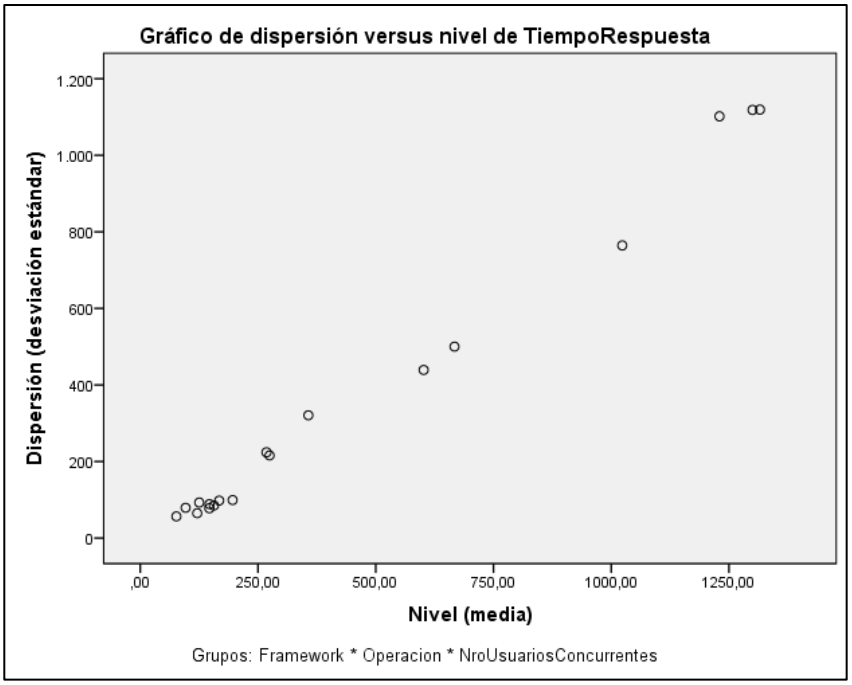


Figura 4.1. Gráfico de Media de factores vs desviación estándar de Tiempo de Respuesta

En las Tablas 4.3 a la 4.8 se presentan los estadísticos descriptivos obtenidos al haber aplicado el análisis de varianza factorial univariante a los datos obtenidos de las pruebas, tomando en cuenta tanto los factores de forma independiente como todas las combinaciones posibles de los mismos, se puede apreciar un resumen de los datos así como las diferencias de medias existentes, las cuales fueron evaluadas posteriormente en la contrastación de hipótesis, a priori se pudo observar una clara diferencia de medias.

Tabla 4.3. Estadísticos de factor *Framework*

Framework	**Media**	**Error estándar**	**Intervalo de confianza al 95%**	
			Límite inferior	**Límite superior**
Spring	339.415	71.774	196.824	482.005
Struts	579.507	71.774	436.916	722.097

En la tabla 4.3 se muestran los estadísticos descriptivos correspondientes al factor *Framework*, mostrando las medias obtenidas para cada uno de sus niveles, y el intervalo de confianza estimado en base a los valores proporcionados en la tabla 4.1.

Tabla 4.4. Estadísticos de factor Operación

Operación	**Media**	**Error estándar**	**Intervalo de confianza al 95%**	
			Límite inferior	**Límite superior**
Autenticación	516.021	87.904	341.384	690.659
Consulta	442.239	87.904	267.601	616.876
Registro	420.122	87.904	245.485	594.759

En la tabla 4.4 se muestran los estadísticos descriptivos correspondientes al factor Operación, mostrando las medias obtenidas para cada uno de sus niveles, y el intervalo de confianza estimado en base a los valores proporcionados en la tabla 4.1.

Tabla 4.5. Estadísticos de factor NroUsuariosConcurrentes

NroUsuariosConcurrentes	**Media**	**Error estándar**	**Intervalo de confianza al 95%**	
			Límite inferior	**Límite superior**
500	128.149	87.904	-46.489	302.786
1000	227.262	87.904	52.624	401.899
1500	1022.972	87.904	848.335	1197.609

En la tabla 4.5 se muestran los estadísticos descriptivos correspondientes al factor *NroUsuariosConcurrentes*, mostrando las medias obtenidas para cada uno de sus niveles, y el intervalo de confianza estimado en base a los valores proporcionados en la tabla 4.1.

Tabla 4.6. Estadísticos de factores *Framework* * Operación

Framework	Operación	Media	Error estándar	Intervalo de confianza al 95%	
				Límite inferior	Límite superior
Spring	Autenticación	448.306	124.315	201.331	695.280
	Consulta	303.483	124.315	56.509	550.458
	Registro	266.456	124.315	19.481	513.430
Struts	Autenticación	583.737	124.315	336.763	830.711
	Consulta	580.994	124.315	334.020	827.968
	Registro	573.789	124.315	326.815	820.763

En la tabla 4.6 se muestran los estadísticos descriptivos correspondientes a la combinación de los factores *Framework* y *Operación*, mostrando las medias obtenidas para cada uno de sus niveles, y el intervalo de confianza estimado en base a los valores proporcionados en la tabla 4.1.

Tabla 4.7. Estadísticos de factores Operación * NroUsuariosConcurrentes

Operación	NroUsuariosConcurrentes	Media	Error estándar	Intervalo de confianza al 95%	
				Límite inferior	Límite superior
Autenticación	500	146.400	152.255	-156.080	448.880
	1000	232.125	152.255	-70.355	534.605
	1500	1169.539	152.255	867.059	1472.020
Consulta	500	126.329	152.255	-176.151	428.810
	1000	251.893	152.255	-50.587	554.374
	1500	948.493	152.255	646.013	1250.974
Registro	500	111.717	152.255	-190.764	414.197
	1000	197.767	152.255	-104.714	500.247
	1500	950.883	152.255	648.403	1253.364

En la tabla 4.7 se muestran los estadísticos descriptivos correspondientes a la combinación de los factores *Operación* y *NroUsuariosConcurrentes*, mostrando las medias obtenidas para cada uno de sus niveles, y el intervalo de confianza estimado en base a los valores proporcionados en la tabla 4.1.

Tabla 4.8. Estadísticos de factores *Framework* * Operación * NroUsuariosConcurrentes

Framework	Operación	NroUsuariosConcurrentes	Media	Error estándar	Intervalo de confianza al 95%	
					Límite inferior	Límite superior
Spring	Autenticación	500	125.300	215.321	-302.472	553.072
		1000	196.233	215.321	-231.539	624.005
		1500	1023.383	215.321	595.611	1451.155
	Consulta	500	96.262	215.321	-331.510	524.034
		1000	146.930	215.321	-280.842	574.702
		1500	667.258	215.321	239.486	1095.030
	Registro	500	76.667	215.321	-351.105	504.439
		1000	120.967	215.321	-306.805	548.739
		1500	601.733	215.321	173.961	1029.505
Struts	Autenticación	500	167.500	215.321	-260.272	595.272
		1000	268.017	215.321	-159.755	695.789
		1500	1315.695	215.321	887.923	1743.467
	Consulta	500	156.397	215.321	-271.375	584.169
		1000	356.857	215.321	-70.915	784.629
		1500	1229.728	215.321	801.956	1657.500
	Registro	500	146.767	215.321	-281.005	574.539
		1000	274.567	215.321	-153.205	702.339
		1500	1300.033	215.321	872.261	1727.805

En la tabla 4.8 se muestran los estadísticos descriptivos correspondientes a la combinación de los factores *Framework*, *Operación* y *NroUsuariosConcurrentes*, mostrando las medias obtenidas para cada uno de sus niveles, y el intervalo de confianza estimado en base a los valores proporcionados en la tabla 4.1.

Cabe resaltar que en los resultados obtenidos existen valores negativos en los intervalos de confianza calculados con un 95%, como por ejemplo en la tabla 4.7 donde se pueden apreciar valores negativos en los límites inferiores, estos valores no representan problema en el tratamiento de los datos realizados, lo que indican estos valores es que una parte de ellos representados bajo la parte del intervalo negativo no tienen significancia estadística por lo que son irrelevantes para el estudio, aclarando que todos los datos que se procesaron, fueron positivos y no existen tiempos de respuesta negativos, la explicación de estos valores se debe a la generación del intervalo de confianza y al error estándar, cuando se calculó, mediante el software SPSS, se calcularon los posibles valores entre los cuales oscilarían los tiempos de respuesta bajo una curva normal (forma de campana).

En la Tabla 4.9 se muestran los resultados de las pruebas inter-sujetos realizadas, así como el diseño factorial de tres factores con sus interacciones, y cuyos datos obtenidos fueron descritos en los párrafos siguientes para lo cual se definió la hipótesis nula y la hipótesis alternativa, siendo:

- H0: No existe diferencia significativa entre los tratamientos.
- H1: Existe diferencia significativa entre los tratamientos.

La fila Modelo corregido recoge todos los efectos del modelo tomados en conjunto, el de cada factor y sus interacciones, el grado de significancia es menor a 0.05, por lo cual se indica que el modelo explica una parte de la variabilidad de la variable dependiente, en concreto 45.2%, el cual resulta de haber dividido la suma de cuadrados del Modelo corregido entre la suma de cuadrados Total corregido.

La fila intersección hace referencia a una constante del modelo y surgió como el producto del número de casos por el cuadrado de la media total de la variable dependiente, esta constante permite contrastar la hipótesis de que la media total de la población es igual a cero, hipótesis rechazada debido al nivel de significancia que posee ($p = 0.000 < 0.05$), lo cual se apreció en la Figura 4.2, mostrando una clara diferencia de medias entre los factores *Framework* y *NroUsuariosConcurrentes* respecto a la variable dependiente *Tiempo de Respuesta*

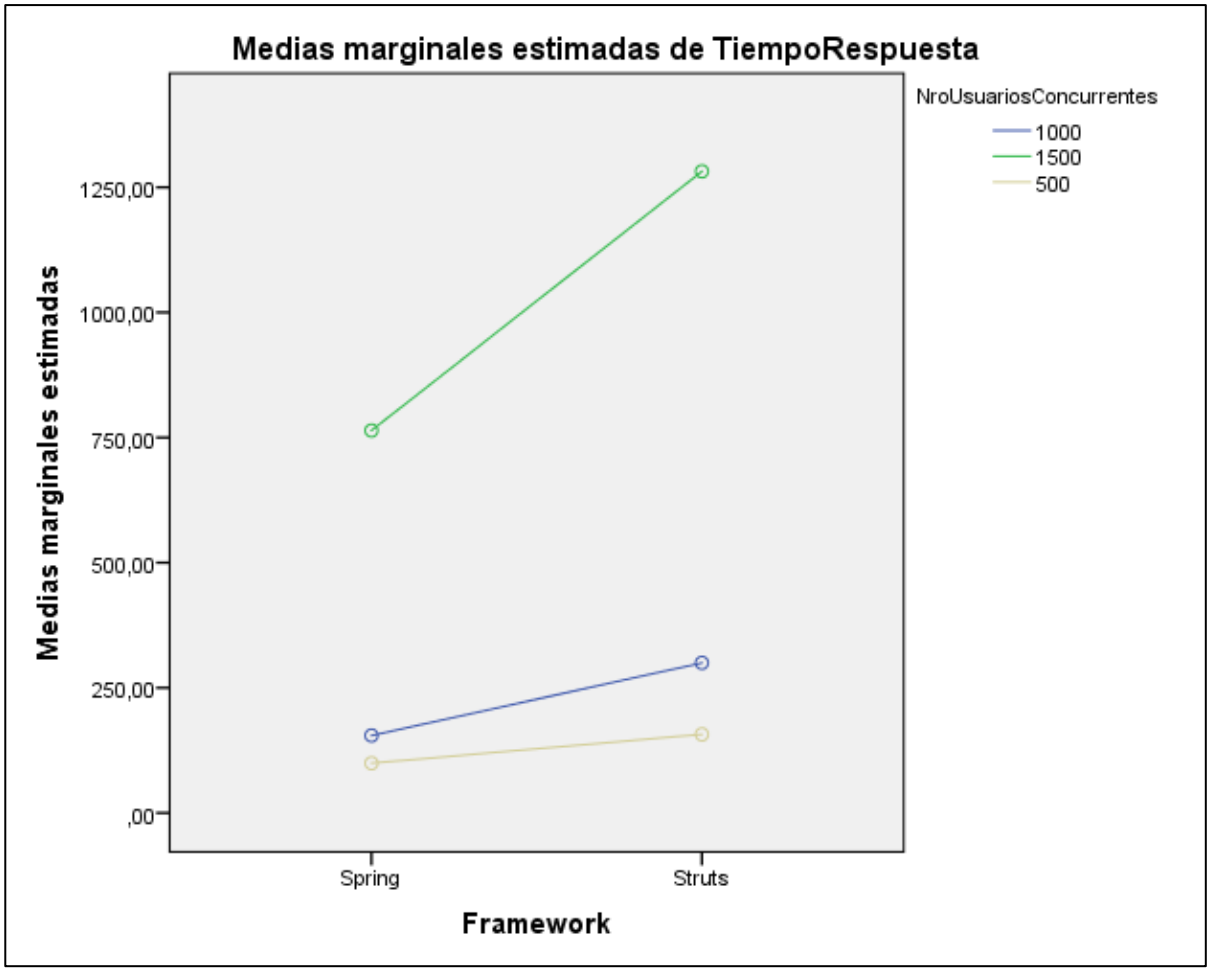


Figura 4.2. Gráfico Media de factor *Framework* vs Media de Tiempo de Respuesta

Las tres filas siguientes muestran los efectos principales de cada uno de los factores que fueron incluidos en el modelo. El factor *Operación* no es significativo ($p = 0.722 > 0.05$).

Las cuatro filas siguientes muestran los efectos de las interacciones entre los factores, donde se obtuvo que los tres factores actúan de forma separada, al no tener un nivel de significancia menor a 0.05.

La fila Error muestra el error residual y la media cuadrática mostrada es una estimación de la varianza de las 18 (2x3x3) poblaciones estudiadas, que se supuso igual en todas ellas.

La fila Total muestra la suma de los cuadrados de las puntuaciones de la variable dependiente y los grados de libertad coincidieron con el tamaño de la muestra (108).

La fila Total corregido muestra la variabilidad de la variable dependiente, es decir, la variación debido al efecto de cada factor, la interacción de los factores y el efecto del error.

El valor de Total es igual a la suma del Modelo corregido más la Intersección más el Error. El Total corregido es la diferencia entre el Total y la Intersección.

En la columna Media cuadrática se observó una diferencia notable en los factores *Framework*, *Operación* y *NroUsuariosConcurrentes*, siendo este factor quien posee una alta variabilidad, mostrando una diferencia de su media cuadrática en aproximadamente 6 millones quinientos mil respecto del segundo factor con la mayor media cuadrática. Esta alta variabilidad afectó directamente a los valores de significancia de los factores que se combinaron con el factor *NroUsuariosConcurrentes*, asimismo, se puede observar en la Figura 4.3, la tendencia que presenta el factor mencionado respecto a sus medias, pronosticando un incremento considerable si se emplearan factores multiplicativos mayores a los empleados en la presente investigación.

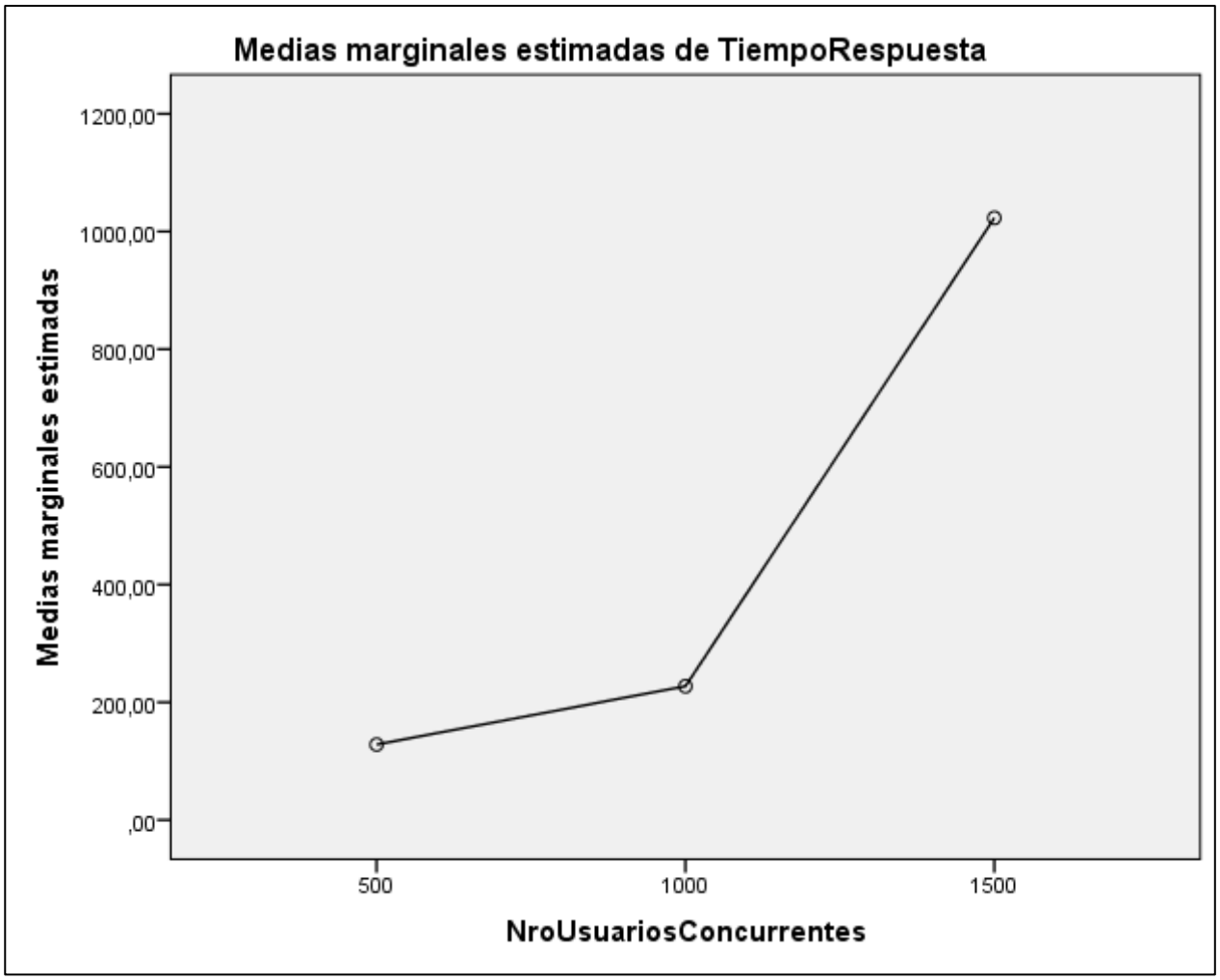


Figura 4.3. Gráfico Media de factor *NroUsuariosConcurrentes* vs Media Tiempo de Respuesta

La columna F es una columna solo informativa, para la presente investigación, debido a que su valor para ser interpretado es necesario contrastarlo con un valor de tabla, esto ya no es necesario, el software SPSS lo realiza y genera la columna Sig, la cual es usada para la interpretación de la tabla, e indicando si se debe aceptar o rechazar la hipótesis nula.

Tabla 4.9. Pruebas de efectos inter-sujetos

Origen	Tipo III de suma de cuadrados	gl	Media cuadrática	F	Sig.
Modelo corregido	20653937,732[a]	17	1214937.514	4.367	0.000
Intersección	22799250.606	1	22799250.606	81.959	0.000
Framework	1556390.628	1	1556390.628	5.595	0.020
Operación	181556.197	2	90778.099	0.326	0.722
NroUsuariosConcurrentes	17324244.502	2	8662122.251	31.139	0.000
Framework * Operación	151877.975	2	75938.988	0.273	0.762
Framework * NroUsuariosConcurrentes	1074900.661	2	537450.330	1.932	0.151
Operación * NroUsuariosConcurrentes	230434.287	4	57608.572	0.207	0.934
Framework * Operación * NroUsuariosConcurrentes	134533.482	4	33633.370	0.121	0.975
Error	25036003.398	90	278177.816		
Total	68489191.737	108			
Total corregido	45689941.131	107			

a. R al cuadrado = 0.452 (R al cuadrado ajustada = 0.349)

Los factores *NroUsuariosConcurrentes* y *Framework*, estadísticamente, son los factores que presentan influencia significativa en la variable dependiente *tiempo de respuesta*, por consiguiente, sería correcto asumir que su combinación debería haber resultado con un valor de significancia menor a 0.05 permitiendo concluir que también poseen influencia juntos, caso que no ocurrió durante el desarrollo de las pruebas estadísticas, pues resultó con un nivel de significancia de 0.151, el motivo por el cual se obtuvo este resultado se debe a la alta variabilidad del factor *NroUsuariosConcurrentes*, como se muestra en la Tabla 4.9 del presente capítulo, la diferencia entre los valores de las medias cuadráticas de los factores que resultaron ser significativos es alta, alrededor de siete millones cien mil, gráficamente se puede observar en la Figura 4.3 y Figura 4.4 del presente capítulo, que ambos factores no presentan una relación, de hecho, ambos presentan un comportamiento creciente, pero la diferencia entre sus medias es notable, como se puede observar sólo en el tramo de la recta entre los valores 1000 y 1500 del factor *NroUsuariosConcurrentes* se puede representar los valores de las medias obtenidas para el factor *Framework*.

El resultado obtenido de ANOVA solo indica que existió una diferencia estadísticamente significativa entre al menos un grupo, no indica el patrón de diferencias existente, es decir, con esta prueba solo se detecta la presencia o ausencia de un efecto global de una o más variables independientes

sobre la variable dependiente, por ello es necesario realizar pruebas a posteriori o post hoc, las cuales consisten en comparar cada par de medias presentes en el estudio, en este caso la comparación se realizó entre las medias de cada uno de los factores del modelo y sus respectivos niveles.

Con la realización de las pruebas post hoc o a posteriori se obtuvieron los datos de las medias que concretamente difieren del resto de medias, como resultado de la ejecución de las pruebas, en la Tabla 4.10 se presenta la evaluación que se obtuvo del factor *Operación*, el cual no presentó un nivel de significancia menor a 0.05 en ningún par de comparación, lo cual se refleja en la Tabla 4.11, donde todos los niveles del factor se encuentran en un mismo subconjunto con un nivel de significancia mayor a 0.05.

Tabla 4.10. Pruebas post hoc – Factor Operación

	(I) Operación	(J) Operación	Diferencia de medias (I-J)	Error estándar	Sig.	Intervalo de confianza al 95%	
						Límite inferior	Límite superior
HSD Tukey	Autenticación	Consulta	73.7828	124.31542	0.824	-222.4735	370.0390
		Registro	95.8992	124.31542	0.721	-200.3571	392.1554
	Consulta	Autenticación	-73.7828	124.31542	0.824	-370.0390	222.4735
		Registro	22.1164	124.31542	0.983	-274.1398	318.3726
	Registro	Autenticación	-95.8992	124.31542	0.721	-392.1554	200.3571
		Consulta	-22.1164	124.31542	0.983	-318.3726	274.1398

Tabla 4.11. Subconjuntos homogéneos – Factor Operación

Operación		N	Subconjunto
			1
HSD Tukey[a,b]	Registro	36	420.1222
	Consulta	36	442.2386
	Autenticación	36	516.0214
	Sig.		0.721
Tukey B[a,b]	Registro	36	420.1222
	Consulta	36	442.2386
	Autenticación	36	516.0214

a. Utiliza el tamaño de la muestra de la media armónica = 36.000.
b. Alfa = .05.

En la Tabla 4.12 se presenta la evaluación que se obtuvo del factor *NroUsuariosConcurrentes*, el cual si presentó un nivel de significancia menor a 0.05 en los pares de comparación, lo cual se refleja en la Tabla 4.12, donde dos de los tres niveles del factor se encuentran en un mismo subconjunto con un nivel de significancia mayor a 0.05, indicando que no hay diferencia significativa entre ellos, mientras que el nivel 1500 se encuentra en el segundo subconjunto, con su nivel de significancia 1, no difiere de sí mismo.

Tabla 4.12. Pruebas post hoc – Factor NroUsuariosConcurrentes

	(I) NroUsuarios Concurrentes	(J) NroUsuariosConcurrentes	Diferencia de medias (I-J)	Error estándar	Sig.	Intervalo de confianza al 95%	
						Límite inferior	Límite superior
HSD Tukey	1000	1500	-795,7103*	124.31542	0.000	-1091.9665	-499.4540
		500	99.1131	124.31542	0.706	-197.1432	395.3693
	1500	1000	795,7103*	124.31542	0.000	499.4540	1091.9665
		500	894,8233*	124.31542	0.000	598.5671	1191.0796
	500	1000	-99.1131	124.31542	0.706	-395.3693	197.1432
		1500	-894,8233*	124.31542	0.000	-1191.0796	-598.5671

*. La diferencia de medias es significativa en el nivel 0.05.

Tabla 4.13. Subconjuntos homogéneos – Factor NroUsuariosConcurrentes

NroUsuariosConcurrentes		N	Subconjunto	
			1	2
HSD Tukey[a,b]	500	36	128.1486	
	1000	36	227.2617	
	1500	36		1022.9719
	Sig.		0.706	1.000
Tukey B[a,b]	500	36	128.1486	
	1000	36	227.2617	
	1500	36		1022.9719

a. Utiliza el tamaño de la muestra de la media armónica = 36.000.
b. Alfa = 0.05.

En las pruebas post hoc mostradas en la Tabla 4.12, tanto el límite superior e inferior del intervalo de confianza son negativos, esto debido a que la diferencia de medias fue realizada entre un valor menor y un valor mayor, mas no representa impedimento para su interpretación.

Las pruebas post hoc no fueron realizadas para el factor *Framework* porque solo contiene dos niveles, las pruebas post hoc realizan comparaciones contra pares, pero siendo no necesaria la realización de las mismas en la presente investigación, el motivo es debido al número reducido de niveles que presenta el factor y a su vez, el efecto que puede generar sobre la variable dependiente puede ser comprobado con la contrastación de hipótesis, pues el análisis tiene como objetivo determinar la diferencia entre los tiempos de respuesta generados por los niveles del factor en mención.

Realizado el análisis de la significancia de factores presentes en el modelo factorial, se procedió a la prueba que permitiría la contrastación de hipótesis, la prueba de diferencia de medias estadísticas, para lo cual se definió la hipótesis nula y la hipótesis alternativa, siendo:

- H0: X1 = X2
- H1: X1 < X2

Donde X1 representa a la media del tiempo de respuesta del *framework* Spring y X2 representa a la media del tiempo de respuesta del *framework* Struts.

Definida la hipótesis alternativa, se observa que se trata de una prueba de cola izquierda, lo cual indica que se deben emplear las fórmulas mostradas en la Tabla 4.14.

Tabla 4.14. Prueba de cola izquierda

H1	SPSS	t>0	t<0
u1 - u2 < 0	p	1-sig/2<α	sig/2<α

Del procesamiento de los datos con el software SPSS se obtuvieron los estadísticos del factor *Framework*, factor que fue empleado en la contrastación, por ser el factor de interés en la presente investigación y son mostrados en la Tabla 4.15, en la cual se puede apreciar de forma directa que existe una diferencia entre las medias de ambos niveles, pero es necesario el análisis estadístico de los datos mostrados para determinar, estadísticamente, si existe o no una diferencia significativa de las medias.

Tabla 4.15. Estadísticos de factor *Framework*

Framework		**N**	**Media**	**Desviación estándar**	**Media de error estándar**
TiempoRespuesta	Spring	54	339.4148	453.03058	61.64965
	Struts	54	579.5067	792.13117	107.79540

Realizada la prueba de diferencia de medias estadísticas, se obtuvieron los datos mostrados en la Tabla 4.16.

Tabla 4.16. Prueba de muestras independientes

		Prueba de Levene de igualdad de varianzas		**Prueba T para la igualdad de medias**						
		F	**Sig.**	**t**	**gl**	**Sig. (bilateral)**	**Diferencia de medias**	**Diferencia de error estándar**	**95% de intervalo de confianza de la diferencia**	
									Inf	**Sup**
Tiempo Respuesta	Se asumen varianzas iguales	8.371	0.005	-1.933	106	0.056	-240.09	124.18	-486.29	6.11
	No se asumen varianzas iguales			-1.933	84.320	0.057	-240.09	124.18	-487.02	6.84

La prueba de Levene indica un valor de significancia de 0.005, se empleó un intervalo de confianza de 95%, por lo tanto el valor de alfa es de 0.05, comparando ambos valores se obtuvo que el valor de significancia es menor que alfa, rechazando el supuesto de igualdad de varianzas, por lo cual se

trabajó con la segunda fila, no es necesario realizar un tratamiento a los datos al no cumplirse la no igualdad de varianzas, debido a que el software SPSS ya lo realiza.

Se obtuvo un t estadístico negativo de -1.933, lo cual indica que se debe emplear la fórmula sig/2<α mostrada en la Tabla 4.14, no se asumen varianzas iguales por lo que el valor de sig 0.057 es el valor de significancia a evaluar, cabe resaltar que es un valor de significancia bilateral, es decir, representa dos colas, motivo por el cual es indicada la división entre dos en la fórmula, el valor de alfa (α) es de 0.05 debido a que se empleó un nivel de confianza del 95%; con los datos mencionados se obtiene para p el valor de 0.0285.

Con la obtención del valor de p se tienen que evaluar dos supuestos:

- Si p < alfa (α), se rechaza H0
- Si p > alfa (α), se acepta H0

El valor de la variable p es de 0.0285, por lo tanto se cumple el primer supuesto, p < alfa (α), lo cual indica que se debe rechazar H0 y aceptar H1.

Esta diferencia significativa puede ser observada de forma gráfica en la Figura 4.4, en la cual se observó una diferencia entre las medias de los niveles del factor *Framework*, lo cual reforzaría el resultado de la prueba estadística.

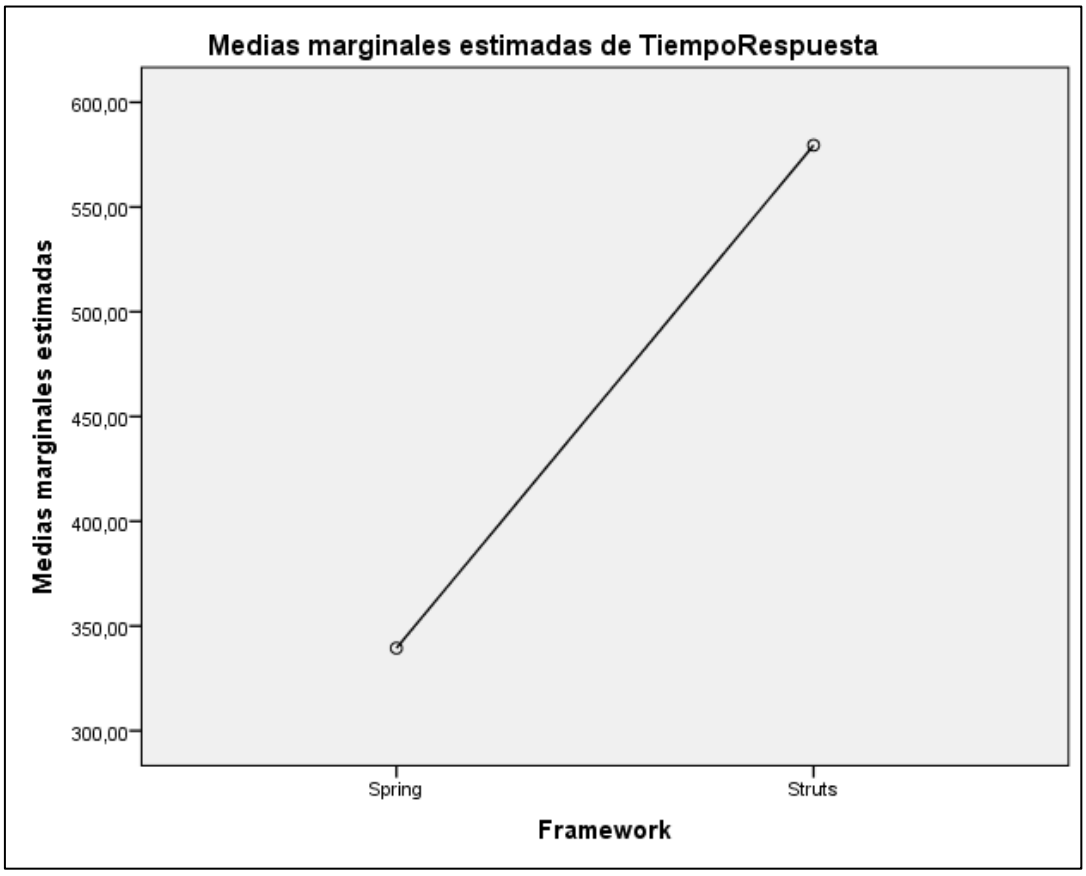


Figura 4.4. Gráfico Media de factor *Framework* vs media de Tiempo de Respuesta

4.2 Discusión

Las pruebas estadísticas empleadas en la presente investigación permitieron determinar que, estadísticamente, dos de tres factores son los que influyen en los tiempos de respuesta, el *número de usuarios concurrentes* con un nivel de significancia de 0.000 y los *framework de lado servidor* con un nivel de significancia de 0.020. El *número de usuarios concurrentes* es una de las métricas empleadas para la evaluación de la performance de aplicaciones Web, como lo indica Watson (2017) respecto a las métricas de performance, al indicar que los *usuarios concurrentes* son una métrica similar a la tasa de solicitudes, pero la considera interesante y que debe ser estudiada. Los *frameworks*, como lo indica Paraschiv (2018), son herramientas que permiten mejorar la *performance* de las aplicaciones o en contraparte, el mal uso de los *frameworks* conlleva a un impacto negativo tanto en las aplicaciones Web como en los clientes, tal como lo indican los estudios realizados por: (a) Walmart Labs (2012), el cual presentaba un estudio realizado sobre el aplicativo Web empleado por las cadenas Walmart, el cual presentaba tiempos de respuesta elevados, determinando que por cada un segundo de mejora en los tiempos de respuesta, el sitio experimentaba hasta un 2% de incremento en su tasa de conversión y por cada 100 milisegundos sus ingresos incrementaban hasta en 1%; (b) Akamai Technologies, Inc (2009), el cual mediante una encuesta realizada a compradores en línea obtuvo que el 40 por ciento de los consumidores esperarán no más de tres segundos para que una página Web cargue antes de abandonar el sitio y (c) Everts (2015), quien determinó que 4.4 segundos era el retraso en los tiempos de respuesta a partir de los cuales una empresa empezaba a registrar pérdidas en sus ingresos por un monto promedio de $4,100 por hora pero a su vez se determinó que la presencia de la lentitud en las aplicaciones Web era diez veces más frecuentes que las interrupciones de servicio. Contrastando con los valores obtenidos en la presente investigación, se observó que tanto la media en los tiempos de respuesta para la aplicación Web desarrollada con Spring (339.4148 ms) y con Struts 2 (579.5067 ms) se encuentran dentro del rango aceptable al no sobrepasar el valor de un segundo.

El desarrollo de aplicaciones Web empleado *frameworks* presenta no solo un efecto en la *performance* de las aplicaciones, sino, un efecto en el proceso de desarrollo de las mismas, es por ello que en la actualidad el uso de *frameworks* es una actividad cotidiana, ya que agilizan los procesos de desarrollo, tal como lo indica Ruiz (2011) en su investigación al comparar el desarrollo de aplicaciones Web de la forma tradicional y mediante la utilización de un *framework*, concluyendo que el desarrollo se lleva a cabo de forma ágil y productiva al emplear *frameworks*, pues brindan una guía adicional de cómo llevar a cabo el desarrollo; o como lo indica Ramírez (2017) en su investigación al concluir que dependiendo de los *frameworks* que se empleen, se pueden presentar diferencias significativas en los tiempos de ejecución, diferencias que pueden generar problemas como los presentados en el párrafo anterior.

La realización de análisis comparativos como los llevados a cabo por Gonzales y Tarifeño (2016), Guerrero (2016) y Peña y Cambisaca (2015), corroboran que los *frameworks* permiten el desarrollo de aplicaciones y/o actividades de forma ordenada, siempre y cuando posean un grado de madurez elevado, y que previamente se hallan realizado las investigaciones correspondientes según los objetivos a cumplir, así como la posibilidad de la adopción de parámetros de comparación basados en la experiencia de empresas relacionadas a las tecnologías de la información, como por ejemplo, ObjectWatch Inc., con el fin de poder determinar de forma correcta si el rendimiento de los *frameworks* investigados afectarán, positiva o negativamente, a las actividades que se planteen realizar o permitan predecir el comportamiento de las aplicaciones desarrolladas.

En la realización de la contrastación de hipótesis se obtuvo como resultado que existe una diferencia estadística significativa en los tiempos de respuesta generados por el *framework* Spring y el *framework* Struts dando por aceptada la hipótesis planteada en la presente investigación, a su vez, se obtuvo que el *framework* que genera los menores tiempos de respuesta es Spring con una media de 339.4148 ms frente a una media de 579.5067 ms perteneciente a Struts 2 (ver Tabla 4.15 y 4.16 para

mayor detalle). Parte de este resultado se debe a la forma en que cada *framework* maneja sus componentes, como lo describe Cygnet Infotech (2015), que indicando las diferencias entre cada *framework*, resaltando el hecho de que Spring mantiene sus controladores a lo largo de las sesiones. Por su contraparte, Struts maneja sus actions (controladores) de una forma distinta, creando instancias para cada petición que esté procesando, generando problemas de rendimiento si se presenta una gran cantidad de peticiones. Otro punto a favor de Spring, mencionado por Poutsma (2017), es que existen distintas mejoras en los módulos del *framework*, buscando asegurar una mejor *performance* ya que de forma general, las aplicaciones desarrolladas bajo Java necesitan ser optimizadas, como lo sugiere Paraschiv (2018), con el fin de evitar problemas cuando la aplicación Web afronte grandes cargas de trabajo. Este resultado refleja lo obtenido en la investigación realizada por Spano (2009) con versiones anteriores de los *frameworks* Spring y Struts 2, 2.5.x y 2.1.6 respectivamente, mostrando que Spring en versiones previas poseía un *performance* superior a Struts y ya definía una estructura modular fija, la cual mantiene en la actualidad incluyendo mejoras sobre los módulos que poseía e integrando nuevas tecnologías (Spring, 2017), por su parte Struts se mantiene como un core mejorando sus características y permitiendo la integración con otros *frameworks* y especificaciones dadas por el estándar JEE (Apache Software Foundation, Inc, 2015).

Como lo estipulan las pruebas estadísticas, Spring presenta un mejor *performance* frente a Struts 2, confirmando lo descrito por Data Flair (2018), quien hace énfasis en las características del *framework*, mostrando una tendencia a las mejoras continuas en busca de una mejor performance, orientando el desarrollo a las pruebas, asimismo, la empresa que tiene a cargo el desarrollo de Spring, tienen por enfoque proveer la mejor experiencia de usuario (Pivotal Software, Inc, 2017). A su vez, existen diversos autores especialistas en Spring, que permiten optimizar las aplicaciones Web para que trabajen en entornos con una alta carga de trabajo, como lo indicado por Mehta, Shah, Shah, Goswami y Radadiya (2018), mostrando diferentes parámetros de configuración y diferentes técnicas que aseguran un incremento en la *performance*. Es por ello que la mayoría de aplicaciones Web desarrolladas con este *framework* pueden obtener buenos resultados al aplicarse algunas de las métricas de *performance* como las planteadas por Arsenault (2017), tales como la tasa de conversión, la latencia, el peso de la propia aplicación Web, etc. Aunque el desarrollo bajo el estándar Java Enterprise Edition (JEE, la especificación oficial para desarrollo de aplicaciones Web bajo el lenguaje Java) actualmente presenta opciones de desarrollo similares a las ofrecidas por Spring, tal como lo comenta Gordeev (2017), presentando la propuesta de JEE como una declaración de guerra hacia Spring y realizando una comparación entre ambas formas de desarrollo, resaltando que JEE en la actualidad ha realizado procesos de mejora en cuanto a *performance*, recomendando emplear tanto Spring como JEE, dependiendo de los conocimientos que se posean, pues ambos presentan cierto reto para las personas que se inician en el desarrollo de aplicaciones Web bajo ambos marcos de trabajo.

Con los resultados de las pruebas estadísticas se puede llevar a cabo una explicación respecto a la relación que se presenta en principio entre los tiempos de respuesta y el uso de un determinado *framework*. Esto debido a que cada *framework* realiza la gestión del flujo de una aplicación Web (Spring, 2017), por lo que dependerá del grado de madurez de un *framework* el realizar una gestión correcta y en parte también dependerá de la habilidad y conocimiento de los desarrolladores que usen los *frameworks*. Pues son los desarrolladores los que deciden si emplean de forma correcta o no los *frameworks*. Si, adicionalmente, se toman en cuenta la cantidad de usuarios concurrentes en una aplicación Web, el comportamiento que tendrán los tiempos de respuesta presentarán variantes complejas de explicar, pues no se podrá determinar en qué medida, cada factor que se agregue al estudio está afectando a los tiempos de respuesta, lo que puede ser afirmado con certeza es que sin importar el *framework* que se emplee, si la cantidad de usuarios concurrentes se incrementa, los tiempos de respuesta siempre mostrarán un incremento (Walmart Labs, 2012). Esto se debe a que cada petición que realiza un usuario a una aplicación Web genera un consumo de recursos en el servidor de aplicaciones y a su vez genera carga que la aplicación debe controlar. Si el número de usuarios que acceden a una aplicación crece, entonces se deberán destinar los recursos necesarios para atender a cada una de las peticiones, esto generará que

los tiempos de respuesta se eleven ya que los recursos son limitados, como se ha mostrado en la presente investigación, al determinar el número de usuarios concurrentes que el servidor de aplicaciones podía manejar. Como se observó, bajo el modelo planteado en la presente investigación solo explica en parte el motivo de la variación de los tiempos de respuesta. Tal como lo indican Ranganath (2018), Quintessence (2018), McPeak (2018), Kumar (2018), Gardner (2018) y Everts (2015) respecto a pruebas de *performance* general, de aplicaciones Web, de servidores e interacción de clientes, los factores que influyen pueden llegar a ser demasiados como para poder ser estudiados en su conjunto. En lugar de ello, se plantean pautas para poder llevar a cabo mediciones, todo ello quedando fuera de los alcances de la presente investigación.

CONCLUSIONES

- Existe diferencia significativa, desde el punto de vista estadístico, en los tiempos de respuesta entre un módulo Web desarrollado con el *framework* Spring y el *framework* Struts 2, siendo el *framework* Spring el que presentó una mejor *performance* (media de 339.42 ms).
- El principal factor que genera la diferencia significativa en los tiempos de respuesta es el *número de usuarios concurrentes* (sig 0.000), relegando a segundo plano el factor *framework de lado servidor* (sig 0.020).
- El factor *tipo de operación* no presenta significancia, desde el punto de vista estadístico (sig 0.722).
- La alta variabilidad del factor: *número de usuarios concurrentes* (media cuadrática 8662122.251), originó que no se llevara a cabo un estudio del efecto que causan la combinación de los factores *framework de lado servidor* y *número de usuarios concurrentes* sobre los tiempos de respuesta.
- Se logró llevar a cabo la medición correcta de los tiempos de respuesta mediante el empleo de las herramientas de software de testeo, guías de observación y los parámetros de desarrollo de los módulos Web.

RECOMENDACIONES

- Realizar estudios comparativos con las futuras versiones de los *frameworks* Spring y Struts, dado que está próxima a ser liberada la versión 3 de Struts anunciando cambios importantes en su *core* y nuevas características.
- Realizar estudios comparativos incluyendo al estándar JEE, pues en sus últimas versiones ha implementado la mayor parte de las funcionalidades que brinda Spring con su contenedor de inyección de dependencias.
- Realizar investigaciones adicionales sobre los tiempos de respuesta máximos y mínimos de ambos *frameworks* para determinar un factor de rendimiento, esto sería de gran utilidad, pues sería un complemento a la presente investigación, permitiendo tomar con un grado mayor de certeza decisiones en el desarrollo de futuras aplicaciones Web.
- Al llevar a cabo test de rendimiento, siempre tener en cuenta la revisión de los archivos de registro de las herramientas en uso y de los servidores de aplicaciones, debido a que la aparición de errores incrementa los tiempos de respuesta y por consiguiente se obtendrán datos erróneos.

REFERENCIAS BIBLIOGRÁFICAS

Adobe Systems Incorporated. (22 de Febrero de 2017). *Aspectos básicos de las aplicaciones web.* Obtenido de https://helpx.adobe.com/es/dreamweaver/using/web-applications.html

Akamai Technologies, Inc. (2009). *Akamai Reveals 2 Seconds As The New Threshold Of Acceptability For ECommerce Web Page Response Times.* Obtenido de https://www.akamai.com/us /en/about/news/press/2009-press/akamai-reveals-2-seconds-as-the-new-threshold-of-acceptability-for-ecommerce-web-page-response-times.jsp

Apache Software Foundation. (2017). *JMeter*. Obtenido de http://jmeter.apache.org/

Apache Software Foundation, Inc. (2015). *Apache Struts 2 Documentation*. Obtenido de https://cwiki. apache.org/confluence/display/WW/Home

Apache Software Foundation, Inc. (2017). *About Us*. Obtenido de https://www.apache.org/

Arsenault, C. (1 de Junio de 2017). *14 Important Website Performance Metrics You Should Be Analyzing.* Obtenido de https://www.keycdn.com/blog/website-performance-metrics/

Bloc. (14 de Diciembre de 2016). *A Comparison of Frontend and Backend Web Development*. Obtenido de https://blog.bloc.io/frontend-vs-backend-web-development/

Brukardt, R. (s.f.). *Execution Time.* Obtenido de http://www.adaic.org/resources/add_content/standards /05rm/html/RM-D-14.html

CA Technologies. (7 de Enero de 2014). *Cómo controlar el número de Servidores de Carga y Usuarios (Hilos Jmeter) en BlazeMeter.* Obtenido de https://www.blazemeter.com/blog/how-control-number-load-servers-and-users-jmeter-threads-blazemeter

Canavos, G. (1992). *Probabilidad y estadística para ingenieros.* México: Mc Graw Hill.

Chauhan, S. (2013). *Understanding Inversion of Control, Dependency Injection and Service Locator.* Obtenido de http://www.dotnettricks.com/learn/dependencyinjection/understanding-inversion-of-control-dependency-injection-and-service-locator

Cygnet Infotech. (15 de Diciembre de 2015). *STRUTS 2 vs SPRINGMVC*. Obtenido de https://www. cygnet-infotech.com/blog/struts-2-vs-springmvc

Data Flair. (21 de Junio de 2018). *Spring Framework Features – Why Spring Framework is Popular*. Obtenido de https://data-flair.training/blogs/spring-framework-features/

Everts, T. (16 de Noviembre de 2015). *A Brief History of Web Performance ROI.* Obtenido de https://dzone.com/articles/a-brief-history-of-web-performance-roi

Gardner, Z. (13 de Agosto de 2018). *Improving Performance in Enterprise Web Applications.* Obtenido de https://dzone.com/articles/improving-performance-in-enterprise-web-applicatio

Gervasio, A. (19 de Diciembre de 2016). *Introducción a Contextos e Injección de Dependencias - CDI.* Obtenido de https://www.sitepoint.com/introduction-contexts-dependency-injection-cdi/

Gonzales Inga, J. P., & Tarifeño Montero, L. M. (2016). *Análisis comparativo de frameworks de arquitectura empresarial para el alineamiento estratégico de tecnologías de información.* Perú: Universidad Señor de Sipán. Obtenido de http://repositorio.uss.edu.pe/xmlui/bitstream/handle/

uss/2692/InformeFinal_Gonzales%20Inga%20Jhonathan%20Pa%c3%bal%20%26%20Tarife%c3%b1o%20Montero%20Lisbeth%20Medalid.pdf

Gordeev, E. (10 de Febrero de 2017). *The on-going war between JEE and Spring framework*. Obtenido de https://www.linkedin.com/pulse/on-going-war-between-jee-spring-framework-evgeni-gordeev

Guerrero Benalcázar, R. (2016). *Estudio comparativo de los frameworks Ruby on Rails y Django para la implementación de un sistema informático de control y administración de network marketing.* Ecuador: Universidad Técnica del Norte. Obtenido de http://repositorio.utn.edu.ec/bitstream/123456789/5356/1/04%20ISC%20414%20TESIS%20DE%20GRADO.pdf

Hernández Sampieri, R., Fernández Collado, C., & Baptista Lucio, M. (2014). *Metodología de la Investigación Científica* (Sexta ed.). México: McGraw - Hill.

International Business Machines Corporation. (1993). *IBM Dictionary of Computing.* Estados Unidos: McGraw-Hill.

Janssen, D., & Janssen, C. (s.f.). *Client/Server Architecture*. Obtenido de https://www.techopedia.com/definition/438/clientserver-architecture

Janssen, D., & Janssen, C. (s.f.). *Object-Relational Mapping*. Obtenido de https://www.techopedia.com/definition/24200/object-relational-mapping--orm

Jboss, Inc. (s.f.). *Hibernate ORM.* Obtenido de http://hibernate.org/orm/

Koirala, S. (2014). *Dependency Injection (DI) vs. Inversion of Control (IOC).* Obtenido de https://www.codeproject.com/Articles/592372/Dependency-Injection-DI-vs-Inversion-of-Control-IO

Kumar Namachivayam, N. (13 de Agosto de 2018). *How to Categorize Performance Testing Defects.* Obtenido de https://dzone.com/articles/how-to-categorize-performance-testing-defects

Lee, D. (7 de Julio de 2015). *The Role of Concurrent Users in Load Testing.* Obtenido de https://www.security.neustar/blog/the-role-of-concurrent-users-in-load-testing

Mandliya, A. (2012). *Introduction to Struts 2.* Obtenido de https://www.java2blog.com/welcome-to-strut2/

McPeak, A. (10 de Agosto de 2018). *Move Fast, Break Things: How to Test the Limits of Your Web App.* Obtenido de https://dzone.com/articles/move-fast-break-things-how-to-test-the-limits-of-y

Mehta, C., Shah, S., Shah, P., Goswami, P., & Radadiya, D. (2018). *Hands-On High Performance with Spring 5: Techniques for scaling and optimizing Spring and Spring Boot applications.* Birmingham, Reino Unido: Packt Publishing Ltd.

Microsoft Developer Network. (2007). *Comprender las pruebas web.* Obtenido de https://msdn.microsoft.com/es-es/library/ms182537(v=vs.90).aspx

Microsoft Developer Network. (2016). *Generar y ejecutar una prueba de rendimiento web codificada.* Obtenido de https://msdn.microsoft.com/es-pe/library/ms182552.aspx

Mills, C. D., & Willee, H. (1 de Septiembre de 2017). *Server-side web frameworks*. Obtenido de https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Web_frameworks

Minitab. (30 de Mayo de 2013). *Regression Analysis: How Do I Interpret R-squared and Assess the Goodness-of-Fit?* Obtenido de http://blog.minitab.com/blog/adventures-in-statistics-2/regression-analysis-how-do-i-interpret-r-squared-and-assess-the-goodness-of-fit

Minitab Inc. (2017). *Temas de apoyo.* Obtenido de https://support.minitab.com/es-mx/minitab/18/nav_d1e16655/

Ohye, M. (2010). *Site Performance For Webmasters.* Obtenido de https://www.youtube.com/watch?v=OpMfx_Zie2g

ONGEI. (1 de Mayo de 2016). *Cero Papel*. Obtenido de http://www.gobiernodigital.gob.pe/cero_papel/cero_papel.asp

Oracle Corporation. (s.f.). *JavaServer Faces Technology Overview.* Obtenido de http://www.oracle.com/technetwork/java/javaee/overview-140548.html

Paraschiv, E. (2 de Enero de 2018). *How to Improve the Performance of a Java Application*. Obtenido de https://stackify.com/java-performance/

Peña Arpi, J. F., & Cambisaca Sánchez, M. G. (2015). *Análisis Comparativo entre los Frameworks Javascript MVC, Angularjs y Ember JS para el Desarrollo de Aplicaciones Web. Caso Práctico: Sistema de Control de Botiquin Veterinario para el MAGAP, Morona Santiago.* Ecuador: Escuela Superior Politécnica de Chimborazo. Obtenido de http://dspace.espoch.edu.ec/bitstream/123456789/4583/1/18T00617.docx

Péraire, C., Edwards, M., Fernandes, A., Mancin, E., & Carroll, K. (2007). The IBM Rational Unified Process for System Z. (Primera). Estados Unidos: International Business Machines Corporation.

Pivotal Software, Inc. (2017). *About Us*. Obtenido de https://pivotal.io/about

Poutsma, A. (4 de Diciembre de 2017). *New in Spring Framework 5.0: Functional Web Framework - Arjen Poutsma.* Obtenido de https://content.pivotal.io/springone-platform-2017/new-in-spring-framework-5-0-functional-web-framework-arjen-poutsma

Quintessence, A. (13 de Agosto de 2018). *Monitoring Server Performance.* Obtenido de https://dzone.com/articles/monitoring-server-performance

Ramirez Coronado, L. G. (2017). *Análisis comparativo del tiempo de ejecución de una aplicación Web desarrollada en diferentes marcos de trabajo en el lado cliente y en el lado servidor.* Piura: Universidad Nacional de Piura.

Ranganath, A. (06 de Agosto de 2018). *3 Trends in Web Performance.* Obtenido de https://dzone.com/articles/3-trends-in-web-performance

Ruiz Robles, J. (2011). *Comparativa entre el desarrollo Web usando el Framework Jboss Seam y el desarrollo tradicional.* Piura: Universidad de Piura. Obtenido de https://pirhua.udep.edu.pe/bitstream/handle/11042/2054/ING_495.pdf

Sahni, S., & Kumar, R. (2004). *Software Development in Java.* Mohali: Unistar Books.

Sánchez, J. (1976). *Estadística básica aplicada.* Bucaramanga: Editora de libros técnicos ltda.

Smedia. (s.f.). *¿Cuánto tiempo demanda el diseño y desarrollo de un sitio Web?* Obtenido de http://www.smedia.com/tiempo-disenio-desarrollo-sitio-web.php

Spano, M. (1 de Agosto de 2009). *Using web application frameworks for the development of information systems.* Obtenido de https://is.cuni.cz/webapps/zzp/download/120008317

Spring. (2017). *Spring Framework.* Obtenido de https://projects.spring.io/spring-framework/

Suárez Silva, E. A. (2011). *Análisis comparativo de los Frameworks EJB3 y ADO.NET Entity framework para el desarrollo de aplicaciones empresariales.* Obtenido de http://bibdigital.epn.edu.ec/bitstream/15000/4295/1/CD-3909.pdf

The American Heritage® Science Dictionary. (s.f.). *website*. Obtenido de http://www.dictionary.com/browse/website

The Apache Software Foundation. (s.f.). *Apache Jmeter - Manual de Usuario.* Obtenido de http://jmeter.apache.org/usermanual/

TIOBE. (2017). *TIOBE Index for October 2017.* Obtenido de https://www.tiobe.com/tiobe-index/

Vicéns, J., Herrarte, A., & Medina, E. (Enero de 2005). *Análisis de la Varianza (ANOVA).* Obtenido de https://uam.es/departamentos/economicas/econapli/anova.pdf

Walmart Labs. (2012). *Performance optimization within Walmart.* Obtenido de http://minus.com/msM8y8nyh#1e

Walpole, R. (1987). *Probabilidad y estadística para ingenieros.* México: Interamericana S.A.

Watson, M. (3 de Julio de 2017). *8 Key Application Performance Metrics & How to Measure Them.* Obtenido de https://stackify.com/application-performance-metrics/

ZeroTurnaround. (2017). *Java Web Frameworks Index.* Obtenido de https://zeroturnaround.com/webframeworksindex/

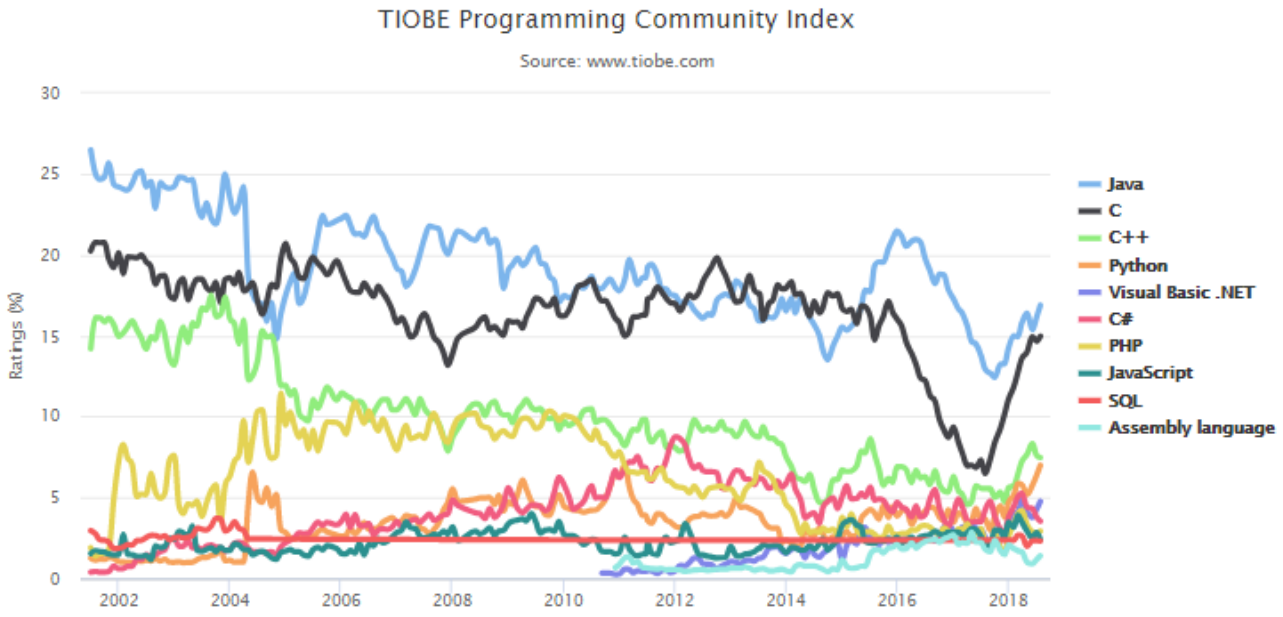
ANEXOS

ANEXO 01. *Ranking* de popularidad - TIOBE Agosto 2018.

Aug 2018	Aug 2017	Change	Programming Language	Ratings	Change
1	1		Java	16.881%	+3.92%
2	2		C	14.966%	+8.49%
3	3		C++	7.471%	+1.92%
4	5	↑	Python	6.992%	+3.30%
5	6	↑	Visual Basic .NET	4.762%	+2.19%
6	4	↓	C#	3.541%	-0.65%
7	7		PHP	2.925%	+0.63%
8	8		JavaScript	2.411%	+0.31%
9	-	⇈	SQL	2.316%	+2.32%
10	14	⇈	Assembly language	1.409%	-0.40%
11	11		Swift	1.384%	-0.44%
12	12		Delphi/Object Pascal	1.372%	-0.45%
13	17	⇈	MATLAB	1.366%	-0.25%
14	18	⇈	Objective-C	1.358%	-0.15%
15	10	⇊	Ruby	1.182%	-0.78%
16	9	⇊	Perl	1.175%	-0.82%
17	16	↓	Go	0.996%	-0.65%
18	15	↓	R	0.965%	-0.80%
19	13	⇊	Visual Basic	0.922%	-0.89%
20	21	↑	PL/SQL	0.702%	-0.51%

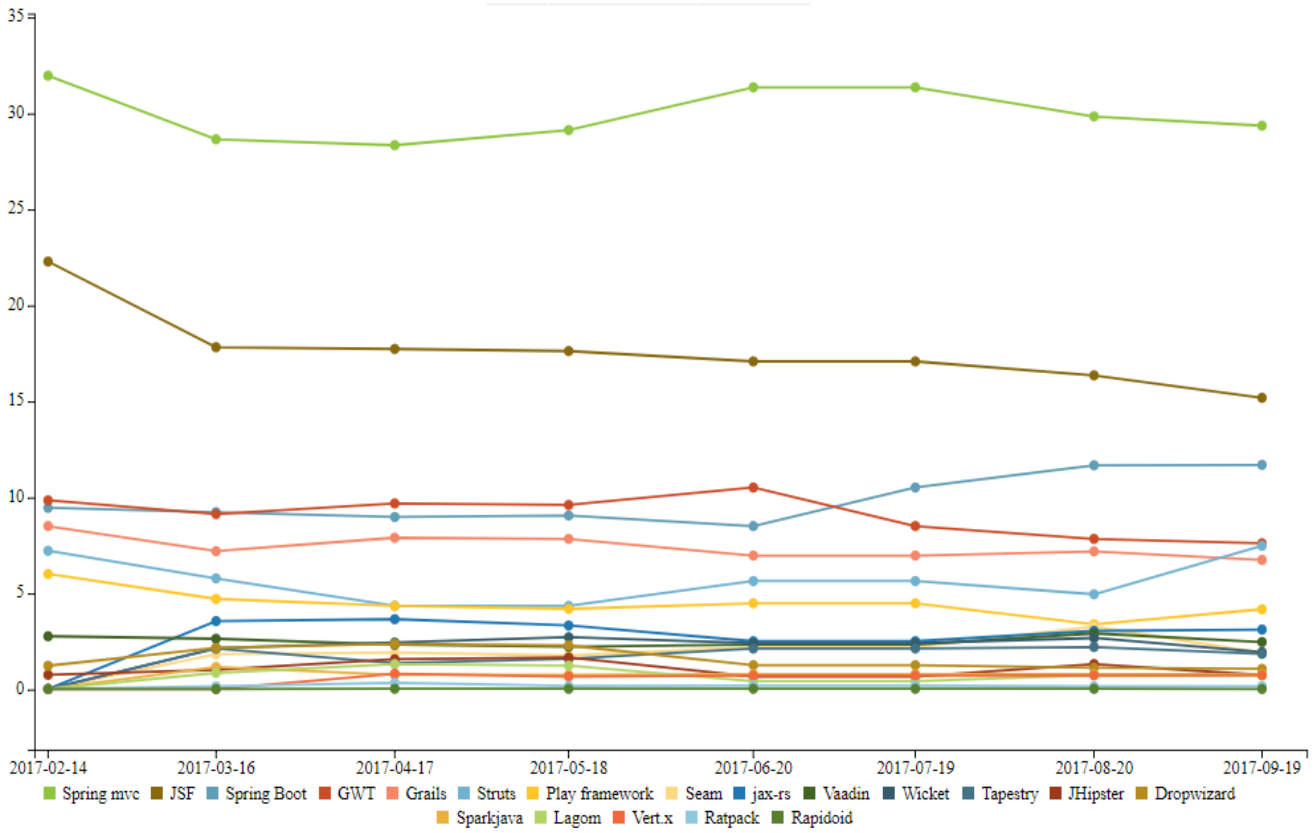
Fuente: TIOBE, 2018

ANEXO 02. Gráfico de tendencias - TIOBE Agosto 2018.



Fuente: TIOBE, 2018

ANEXO 03. Gráfico de tendencias - *ZeroTurnaround* Septiembre 2017.



Fuente: *ZeroTurnaround*, 2017

ANEXO 04. Guía de observación N°01

TIEMPOS DE RESPUESTA DEL MÓDULO MÁS DESARROLLADO EN LA CIUDAD DE PIURA E IMPLEMENTADO CON DISTINTAS COMBINACIONES DE FRAMEWORKS WEB.

La presente guía de observación es utilizada con la intención de medir los tiempos de cada una de las combinaciones de frameworks seleccionados para el desarrollo de los módulos Webs, lo cual proporcionará la información necesaria para definir la combinación de framework que dan menor tiempo de repuesta.

El observador deberá registrar el tiempo respuesta, expresado en milisegundos de cada prueba para cada combinación.

	Factor usuarios								
	C1			**C2**			**C3**		
	Factor operaciones								
	Autenticación	Registro	Consultas	Autenticación	Registro	Consultas	Autenticación	Registro	Consultas
Spring									
Struts 2									

C1: Números de usuarios que normalmente concurren al módulo Web en un caso real.

C2: El doble de número de usuarios de C1.

C3: El triple de número de usuarios de C1.

Fecha de inicio de la observación: ____/____ /_____ (dd/mm/aaaa)

Fecha de finalización de la observación: ____/____ /_____ (dd/mm/aaaa)

Observaciones:

_____

_____

_____

Investigador:

_____

Fecha de entrega: ____ /_____/_____ (dd/mm/aaaa)

Verificado ☐ Archivado ☐ Entregado ☐

ANEXO 05. Validación de instrumentos.

A continuación, se anexan las constancias de validación de la guía de observación; ya firmadas por docentes del departamento de Investigación de Operaciones de la facultad de Industrial e ingenieros de la Escuela de Ingeniería Informática de la Universidad Nacional de Piura.

UNIVERSIDAD NACIONAL DE PIURA
FACULTAD DE INGENIERÍA INDUSTRIAL
ESCUELA PROFESIONAL DE INGENIERÍA INFORMÁTICA

CONSTANCIA DE VALIDACIÓN

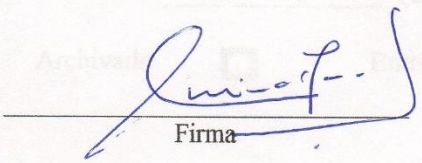
Yo, Jonathan Nima Ramos, identificado con DNI Nº 42627674, de profesión Ingeniero Informatico, con grado de Dr. ejerciendo actualmente como Docente Universitario / Sistemas, en la Institución UNP.

Por medio de la presente hago constar que he revisado con fines de validación de la guía de observación, a los efectos de su aplicación de medir los tiempos de cada una de las combinaciones de frameworks seleccionados para el desarrollo de los módulos Webs, lo cual proporcionará la información necesaria para definir la combinación de framework que da el menor tiempo de repuesta.

Luego de hacer las observaciones pertinentes, puedo formular las siguientes apreciaciones.

ITEMS	CRITERIOS	DEFICIENTE	ACEPTABLE	BUENO	EXCELENTE
Claridad	Está formulado en un lenguaje apropiado			✓	
Objetividad	Está expresado en conductas observables			✓	
Coherencia	Entre los indicadores			✓	
Metodología	Responde al propósito			✓	

En Piura, a los 11 días del mes de Diciembre de 2017.



Firma

UNIVERSIDAD NACIONAL DE PIURA
FACULTAD DE INGENIERÍA INDUSTRIAL
ESCUELA PROFESIONAL DE INGENIERÍA INFORMÁTICA

CONSTANCIA DE VALIDACIÓN

Yo, Anthony Paul Tavara Ramos, identificado con DNI Nº 40734283, de profesión Ing. Informatico, con grado de Magister en Informatica ejerciendo actualmente como Desarrollador de Aplicaciones, en la Institución Universidad Nacional de Piura.

Por medio de la presente hago constar que he revisado con fines de validación de la guía de observación, a los efectos de su aplicación de medir los tiempos de cada una de las combinaciones de frameworks seleccionados para el desarrollo de los módulos Webs, lo cual proporcionará la información necesaria para definir la combinación de framework que da el menor tiempo de repuesta.

Luego de hacer las observaciones pertinentes, puedo formular las siguientes apreciaciones.

ITEMS	CRITERIOS	DEFICIENTE	ACEPTABLE	BUENO	EXCELENTE
Claridad	Está formulado en un lenguaje apropiado			✓	
Objetividad	Está expresado en conductas observables			✓	
Coherencia	Entre los indicadores			✓	
Metodología	Responde al propósito			✓	

En Piura, a los 11 días del mes de Diciembre de 2017.

Firma

UNIVERSIDAD NACIONAL DE PIURA
FACULTAD DE INGENIERÍA INDUSTRIAL
ESCUELA PROFESIONAL DE INGENIERÍA INFORMÁTICA

CONSTANCIA DE VALIDACIÓN

Yo, Carlos Coello Oblitas, identificado con DNI Nº 02624196, de profesión Ing. Industrial, con grado de Magister ejerciendo actualmente como Docente, en la Institución Univ. Nacional de Piura.

Por medio de la presente hago constar que he revisado con fines de validación de la guía de observación, a los efectos de su aplicación de medir los tiempos de cada una de las combinaciones de frameworks seleccionados para el desarrollo de los módulos Webs, lo cual proporcionará la información necesaria para definir la combinación de framework que da el menor tiempo de repuesta.

Luego de hacer las observaciones pertinentes, puedo formular las siguientes apreciaciones.

ITEMS	CRITERIOS	DEFICIENTE	ACEPTABLE	BUENO	EXCELENTE
Claridad	Está formulado en un lenguaje apropiado			✓	
Objetividad	Está expresado en conductas observables				✓
Coherencia	Entre los indicadores				✓
Metodología	Responde al propósito			✓	

En Piura, a los 11 días del mes de Diciembre de 2017.

Carlos Coello

Firma

ANEXO 06. Buenas prácticas en el uso de la herramienta JMeter

Para la obtención de una performance adecuada al emplear la herramienta JMeter, sus creadores han puesto a disposición de los usuarios las buenas prácticas que deben ser tomadas en cuenta:

- Siempre emplear la última versión de JMeter, pues en cada versión se corrigen errores que pueden afectar su uso.
- Emplear el número correcto de hilos, considerando las capacidades de software y hardware con las que se cuentan.
- Al emplear el elemento HTTP(S) Test Script Recorder, siempre emplear filtros, para evitar guardar datos sobre peticiones o archivos que no son de interés para el desarrollo de las pruebas.
- Emplear las variables de usuario definidas dentro de JMeter para generar distintos valores dentro de las pruebas, como por ejemplo, nombres de usuarios.
- Siempre emplear el modo consola de JMeter si se ejecutarán planes de prueba "pesados".
- Evitar el uso de los elementos "View Results Tree" o "View Results in Table" durante la ejecución de las pruebas.
- Emplear archivos CSV cuando se necesiten realizar peticiones con datos personalizados.
- Emplear una salida en formato CSV y no una salida XML.
- Solo guardar los datos que se necesiten.
- Evitar ejecutar las pruebas desde el ordenador que aloja las aplicaciones a revisar.
- Emplear el modo de ejecución de pruebas distribuidas cuando se tenga un elevado número de usuarios concurrentes al momento de realizar las pruebas.

ANEXO 07. Guía de observación – Ejecución de primera prueba

TIEMPOS DE RESPUESTA DEL MÓDULO MÁS DESARROLLADO EN LA CIUDAD DE PIURA E IMPLEMENTADO CON DISTINTAS COMBINACIONES DE FRAMEWORKS WEB.

La presente guía de observación es utilizada con la intención de medir los tiempos de cada una de las combinaciones de frameworks seleccionados para el desarrollo de los módulos Webs, lo cual proporcionará la información necesaria para definir la combinación de framework que dan menor tiempo de repuesta.

El observador deberá registrar el tiempo respuesta, expresado en milisegundos de cada prueba para cada combinación.

	Factor usuarios								
	C1			C2			C3		
	Factor operaciones								
	Autenticación	Registro	Consultas	Autenticación	Registro	Consultas	Autenticación	Registro	Consultas
Spring	5.40	5.40	3.61	9.90	6.70	8.02	11.50	9.70	11.30
Struts 2	5.00	5.60	4.34	4.90	5.30	5.66	38.90	39.50	44.98

C1: Números de usuarios que normalmente concurren al módulo Web en un caso real.

C2: El doble de número de usuarios de C1.

C3: El triple de número de usuarios de C1.

Fecha de inicio de la observación: 10 / 07 / 2018 (dd/mm/aaaa)

Fecha de finalización de la observación: 10 / 07 / 2018 (dd/mm/aaaa)

Observaciones:

C1 : 500 , C2 : 1000 , C3 : 1500
Primera prueba.

Investigador:
Luis Alberto Sandoval Arévalo

Fecha de entrega: 10 / 07 / 2018 (dd/mm/aaaa)

Verificado ☑ Archivado ☑ Entregado ☑

45

ANEXO 08. Guía de observación – Ejecución de segunda prueba

TIEMPOS DE RESPUESTA DEL MÓDULO MÁS DESARROLLADO EN LA CIUDAD DE PIURA E IMPLEMENTADO CON DISTINTAS COMBINACIONES DE FRAMEWORKS WEB.

La presente guía de observación es utilizada con la intención de medir los tiempos de cada una de las combinaciones de frameworks seleccionados para el desarrollo de los módulos Webs, lo cual proporcionará la información necesaria para definir la combinación de framework que dan menor tiempo de repuesta.

El observador deberá registrar el tiempo respuesta, expresado en milisegundos de cada prueba para cada combinación.

	Factor usuarios								
	C1			**C2**			**C3**		
	Factor operaciones								
	Autenticación	Registro	Consultas	Autenticación	Registro	Consultas	Autenticación	Registro	Consultas
Spring	6.40	4.70	3.84	160.50	84.70	92.91	120.40	61.80	66.09
Struts 2	135.10	131.30	135.66	349.50	345.80	363.27	195.10	197.50	207.64

C1: Números de usuarios que normalmente concurren al módulo Web en un caso real.

C2: El doble de número de usuarios de C1.

C3: El triple de número de usuarios de C1.

Fecha de inicio de la observación: 10 / 07 / 2018 (dd/mm/aaaa)

Fecha de finalización de la observación: 10 / 07 / 2018 (dd/mm/aaaa)

Observaciones:

C1 = 500, C2 = 1000, C3 = 1500
Segunda Prueba.

Investigador:
Luis Alberto Sandoval Arévalo

Fecha de entrega: 10 / 07 / 2018 (dd/mm/aaaa)

Verificado ☑ Archivado ☑ Entregado ☑

45

ANEXO 09. Guía de observación – Ejecución de tercera prueba

TIEMPOS DE RESPUESTA DEL MÓDULO MÁS DESARROLLADO EN LA CIUDAD DE PIURA E IMPLEMENTADO CON DISTINTAS COMBINACIONES DE FRAMEWORKS WEB.

La presente guía de observación es utilizada con la intención de medir los tiempos de cada una de las combinaciones de frameworks seleccionados para el desarrollo de los módulos Webs, lo cual proporcionará la información necesaria para definir la combinación de framework que dan menor tiempo de repuesta.

El observador deberá registrar el tiempo respuesta, expresado en milisegundos de cada prueba para cada combinación.

	Factor usuarios								
	C1			C2			C3		
	Factor operaciones								
	Autenticación	Registro	Consultas	Autenticación	Registro	Consultas	Autenticación	Registro	Consultas
Spring	169.90	92.00	89.36	263.40	138.60	150.09	1694.40	856.40	874.45
Struts 2	237.40	237.80	245.57	57.40	61.20	83.61	3105.57	3061.90	3041.00

C1: Números de usuarios que normalmente concurren al módulo Web en un caso real.

C2: El doble de número de usuarios de C1.

C3: El triple de número de usuarios de C1.

Fecha de inicio de la observación: 10 / 07 / 2018 (dd/mm/aaaa)

Fecha de finalización de la observación: 10 / 07 / 2018 (dd/mm/aaaa)

Observaciones:

C1 = 500, C2 = 1000, C3 = 1500
Tercera prueba.
Se repitió prueba debido a problemas con el servidor de aplicaciones.

Investigador:
Luis Alberto Sandoval Arévalo

Fecha de entrega: 10 / 07 / 2018 (dd/mm/aaaa)

Verificado ☑ Archivado ☑ Entregado ☑

45

ANEXO 10. Guía de observación – Ejecución de cuarta prueba

TIEMPOS DE RESPUESTA DEL MÓDULO MÁS DESARROLLADO EN LA CIUDAD DE PIURA E IMPLEMENTADO CON DISTINTAS COMBINACIONES DE FRAMEWORKS WEB.

La presente guía de observación es utilizada con la intención de medir los tiempos de cada una de las combinaciones de frameworks seleccionados para el desarrollo de los módulos Webs, lo cual proporcionará la información necesaria para definir la combinación de framework que dan menor tiempo de repuesta.

El observador deberá registrar el tiempo respuesta, expresado en milisegundos de cada prueba para cada combinación.

	Factor usuarios								
	C1			C2			C3		
	Factor operaciones								
	Autenticación	Registro	Consultas	Autenticación	Registro	Consultas	Autenticación	Registro	Consultas
Spring	189.50	113.70	165.80	265.10	153.10	176.95	1529.00	880.40	968.05
Struts 2	178.80	149.70	179.45	261.50	260.90	374.55	1669.30	1766.90	1572.43

C1: Números de usuarios que normalmente concurren al módulo Web en un caso real.

C2: El doble de número de usuarios de C1.

C3: El triple de número de usuarios de C1.

Fecha de inicio de la observación: 24 / 07 / 2018 (dd/mm/aaaa)

Fecha de finalización de la observación: 24 / 07 / 2018 (dd/mm/aaaa)

Observaciones:

C1 = 500, C2 = 1000, C3 = 1500
Cuarta prueba.

Investigador: Luis Alberto Sandoval Aréstegui

Fecha de entrega: 24 / 07 / 2018 (dd/mm/aaaa)

Verificado ☑ Archivado ☑ Entregado ☑

45

ANEXO 11. Guía de observación – Ejecución de quinta prueba

TIEMPOS DE RESPUESTA DEL MÓDULO MÁS DESARROLLADO EN LA CIUDAD DE PIURA E IMPLEMENTADO CON DISTINTAS COMBINACIONES DE FRAMEWORKS WEB.

La presente guía de observación es utilizada con la intención de medir los tiempos de cada una de las combinaciones de frameworks seleccionados para el desarrollo de los módulos Webs, lo cual proporcionará la información necesaria para definir la combinación de framework que dan menor tiempo de repuesta.

El observador deberá registrar el tiempo respuesta, expresado en milisegundos de cada prueba para cada combinación.

	Factor usuarios								
	C1			C2			C3		
	Factor operaciones								
	Autenticación	Registro	Consultas	Autenticación	Registro	Consultas	Autenticación	Registro	Consultas
Spring	190.50	123.90	127.82	253.60	159.50	185.45	1620.90	879.80	1178.64
Struts 2	291.90	179.70	221.02	627.20	595.20	917.64	1534.70	1610.00	1588.32

C1: Números de usuarios que normalmente concurren al módulo Web en un caso real.

C2: El doble de número de usuarios de C1.

C3: El triple de número de usuarios de C1.

Fecha de inicio de la observación: 24 / 07 / 2018 (dd/mm/aaaa)

Fecha de finalización de la observación: 24 / 07 / 2018 (dd/mm/aaaa)

Observaciones:

C1=500, C2=1000, C3=1500
Quinta prueba.
Se repitió debido a interrupción en el suministro eléctrico.

Investigador:
Luis Alberto Sandoval Arévalo.

Fecha de entrega: 24 / 07 / 2018 (dd/mm/aaaa)

Verificado ☑ Archivado ☑ Entregado ☑

45

ANEXO 12. Guía de observación – Ejecución de sexta prueba

TIEMPOS DE RESPUESTA DEL MÓDULO MÁS DESARROLLADO EN LA CIUDAD DE PIURA E IMPLEMENTADO CON DISTINTAS COMBINACIONES DE FRAMEWORKS WEB.

La presente guía de observación es utilizada con la intención de medir los tiempos de cada una de las combinaciones de frameworks seleccionados para el desarrollo de los módulos Webs, lo cual proporcionará la información necesaria para definir la combinación de framework que dan menor tiempo de repuesta.

El observador deberá registrar el tiempo respuesta, expresado en milisegundos de cada prueba para cada combinación.

	Factor usuarios								
	C1			C2			C3		
	Factor operaciones								
	Autenticación	Registro	Consultas	Autenticación	Registro	Consultas	Autenticación	Registro	Consultas
Spring	190.10	120.30	187.14	224.90	183.20	268.16	1164.10	922.30	1005.02
Struts 2	156.80	176.50	152.34	307.60	389.00	396.41	1350.60	1124.40	924.00

C1: Números de usuarios que normalmente concurren al módulo Web en un caso real.

C2: El doble de número de usuarios de C1.

C3: El triple de número de usuarios de C1.

Fecha de inicio de la observación: 24 / 07 / 2018 (dd/mm/aaaa)

Fecha de finalización de la observación: 24 / 07 / 2018 (dd/mm/aaaa)

Observaciones:

C1 = 500, C2 = 1000, C3 = 1500

Sexta prueba

Investigador: Luis Alberto Sandoval Arévalo

Fecha de entrega: 24 / 07 / 2018 (dd/mm/aaaa)

Verificado ☑ Archivado ☑ Entregado ☑

45

ANEXO 13. Ejemplo de reporte generado por JMeter

Load Test Results

Date report: 2018/07/10

Designed for use with JMeter and Ant.

Summary

# Samples	Failures	Success Rate	Average Time	Min Time	Max Time
58000	0	100.00%	3 ms	0 ms	1321 ms

Pages

URL	# Samples	Failures	Success Rate	Average Time	Min Time	Max Time	
13 /TDSpring	500	0	100.00%	4 ms	1 ms	196 ms	−

Details for Page "13 /TDSpring"

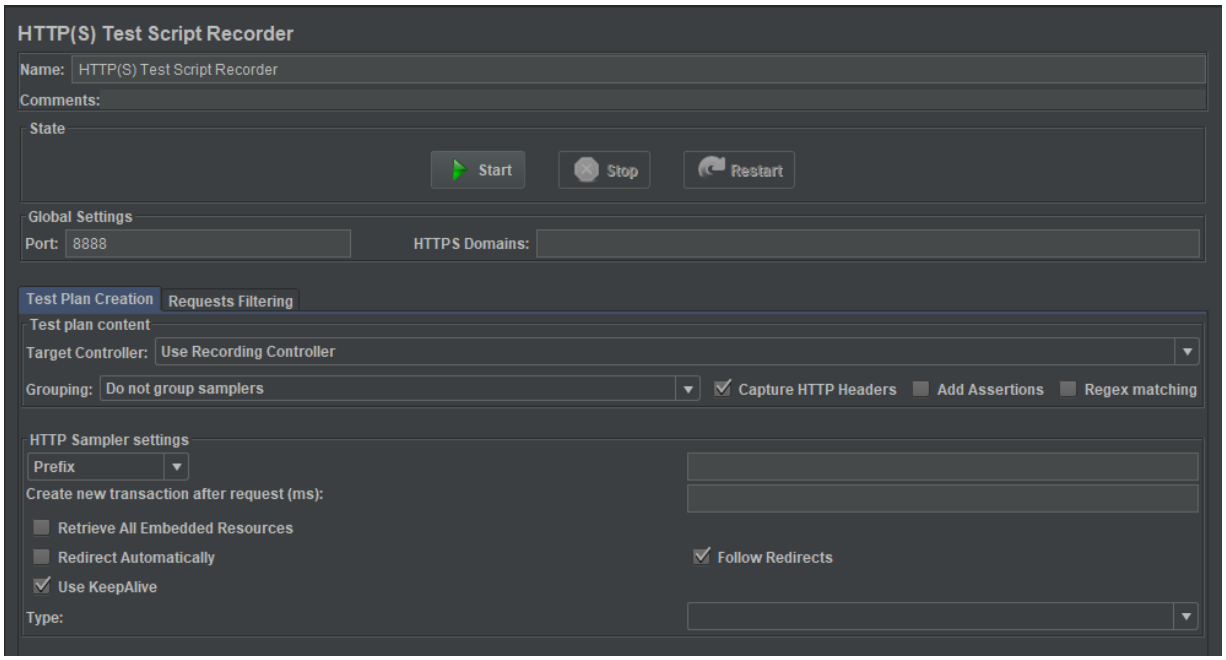
Thread	Iteration	Time (milliseconds)	Bytes	Success
Thread Group 1-1	1	118	7335	true
Thread Group 1-2	2	83	7335	true
Thread Group 1-3	3	21	7335	true
Thread Group 1-6	4	20	7335	true
Thread Group 1-5	5	31	7335	true
Thread Group 1-4	6	9	7335	true
Thread Group 1-8	7	3	7335	true
Thread Group 1-7	8	6	7335	true
Thread Group 1-9	9	6	7335	true
Thread Group 1-10	10	81	7335	true
Thread Group 1-12	11	24	7335	true
Thread Group 1-11	12	34	7335	true
Thread Group 1-13	13	34	7335	true
Thread Group 1-14	14	4	7335	true
Thread Group 1-15	15	7	7335	true
Thread Group 1-16	16	23	7335	true
Thread Group 1-17	17	6	7335	true
Thread Group 1-18	18	15	7335	true
Thread Group 1-19	19	13	7335	true
Thread Group 1-20	20	25	7335	true
Thread Group 1-21	21	7	7335	true
Thread Group 1-22	22	3	7335	true
Thread Group 1-23	23	6	7335	true
Thread Group 1-24	24	2	7335	true
Thread Group 1-25	25	6	7335	true
Thread Group 1-26	26	4	7335	true

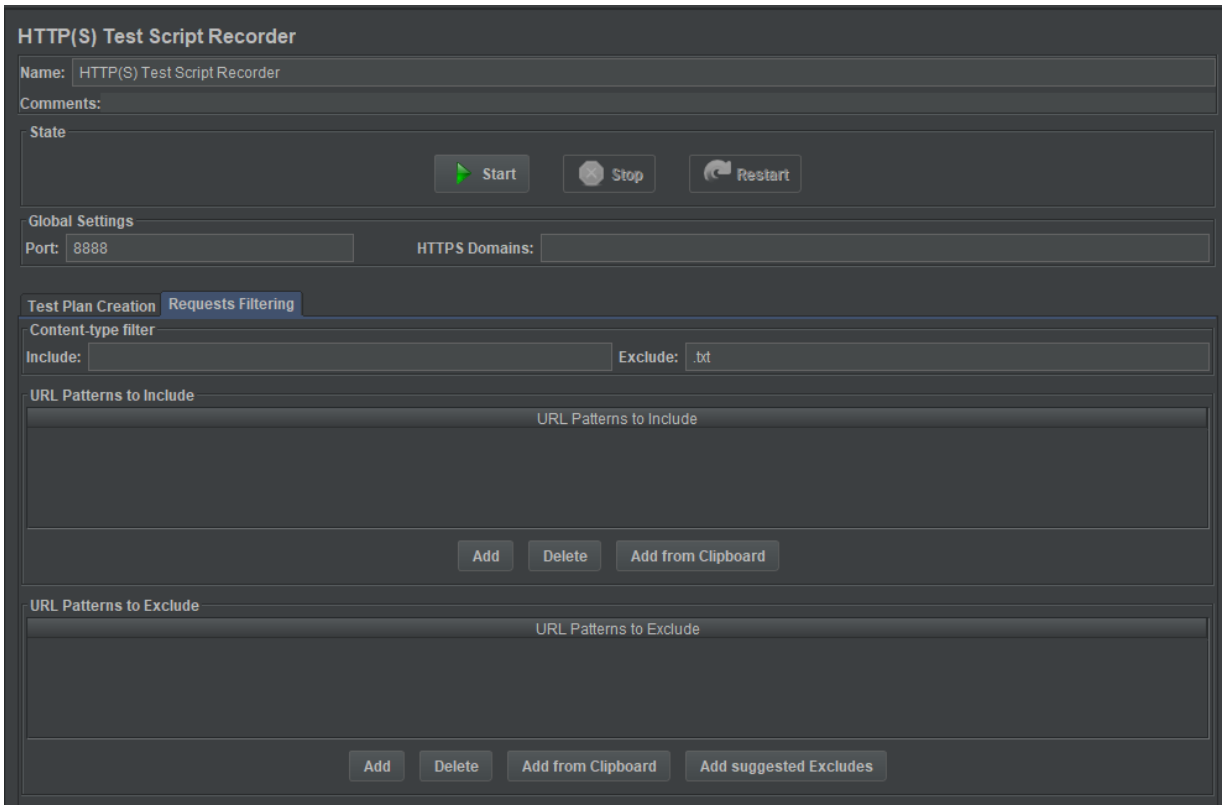
ANEXO 14. Matriz general de consistencia

Título: Análisis comparativo de los tiempos de respuesta de un módulo Web de trámite documentario desarrollado con los frameworks Spring y Struts 2 PIURA-PERÚ. 2018.
Nombre del Tesista: Luis Alberto Sandoval Arévalo

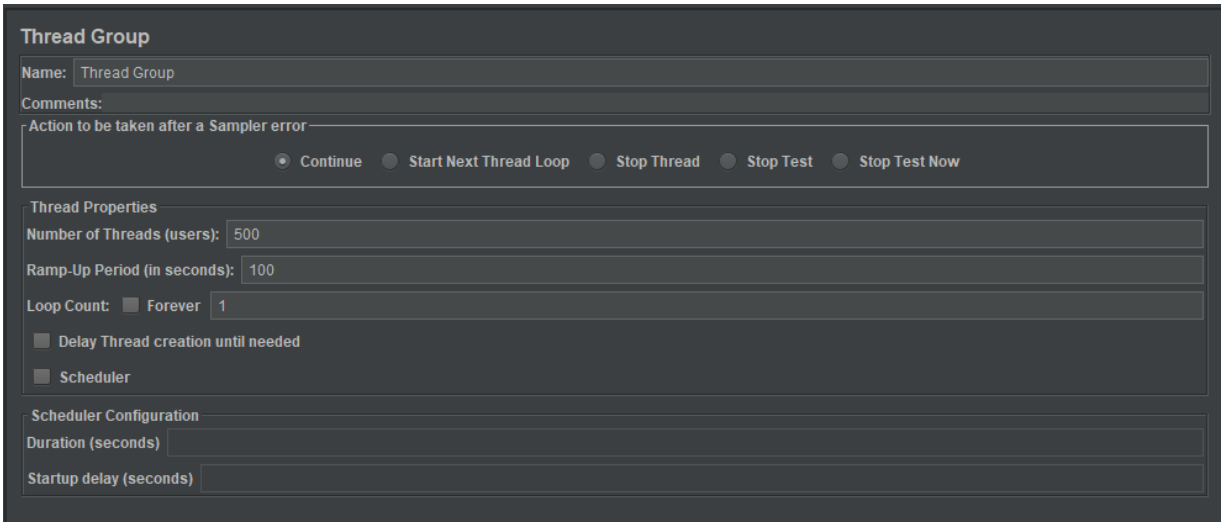
Problemas	Objetivos	Hipótesis	Variables / Indicadores	Metodología
General ¿Cómo se analiza comparativamente los tiempos de respuesta de un módulo Web de trámite documentario desarrollado con los frameworks Spring y Struts 2? **Específicos** 1. ¿Cuáles serán los parámetros necesarios para el desarrollo del módulo Web según la metodología RUP? 2. ¿Cómo medir el tiempo de respuesta del módulo Web de trámite documentario desarrollado con el framework Spring? 3. ¿Cómo medir el tiempo de respuesta del módulo Web de trámite documentario desarrollado con el framework Struts 2?	**General** Analizar comparativamente los tiempos de respuesta de un módulo Web de trámite documentario desarrollado con los frameworks Spring y Struts 2 **Específicos** 1. Determinar los parámetros necesarios para el desarrollo del módulo Web según la metodología RUP. 2. Medir el tiempo de respuesta del módulo Web de trámite documentario desarrollado con el framework Spring. 3. Medir el tiempo de respuesta del módulo Web de trámite documentario desarrollado con el framework Struts 2.	**General** Existe diferencia significativa en el tiempo de respuesta de los módulos Web desarrollados con los frameworks Spring y Struts 2 **Justificación** La decisión de realizar este análisis comparativo, se debe a que la mayoría de organizaciones y desarrolladores suelen basarse en la popularidad y/o fuentes no confiables para elegir el o los frameworks que serán usados para el desarrollo del software, con esta investigación se determinará la mejor opción a elegir entre los frameworks back-end Java más populares para la versión Enterprise Edition (JEE) que permita manejar tiempos de respuesta aceptables. Además, los frameworks están en constante evolución, por lo cual, investigaciones como la presente son necesarias para encontrar las mejores opciones de desarrollo y lograr obtener software de calidad. **Importancia** Con el estudio comparativo se busca facilitar la decisión de usar un determinado framework para el desarrollo de software, esto debido a que cada uno de ellos tiene sus particularidades. Permitirá obtener documentación actualizada y de confianza, la misma que puede ser usada como guía o referencia no solo por profesionales, sino también por alumnos y docentes de los diferentes centros de formación, garantizando un conocimiento más profundo respecto a los frameworks y su impacto en los tiempos de respuesta al ser usados en el desarrollo de aplicaciones Web. Además, servirá como motivación para estudiantes interesados en investigar las tecnologías nacientes, aportará nuevos conocimientos y permitirá evaluar qué tan flexible puede llegar a ser un framework al ser usado en el desarrollo de aplicaciones.	**Unidad de análisis:** Tiempo de respuesta. **Variable:** Usuarios concurrentes **Dimensiones:** Usuarios concurrentes base **Indicadores:** Número de usuarios concurrentes **Variable:** Tiempo de respuesta **Dimensiones:** - Tiempos de respuesta del módulo Web desarrollado con Spring. - Tiempos de respuesta del módulo Web desarrollado con Struts 2. **Indicadores:** - Tiempo de respuesta promedio	**Enfoque:** Cuantitativo **Diseño:** Experimental **Nivel:** Explicativo **Tipo:** Aplicada tecnológica **Técnicas e instrumentos** **De recolección de datos:** - Guía de Observación - Apache JMeter **De procesamiento de datos:** - SPSS - Microsoft Excel **De análisis:** - Análisis de varianza factorial – ANOVA - Diferencia de medias estadísticas

ANEXO 15. HTTP(S) test script recorder





ANEXO 16. Thread group



ANEXO 17. Elementos de configuración